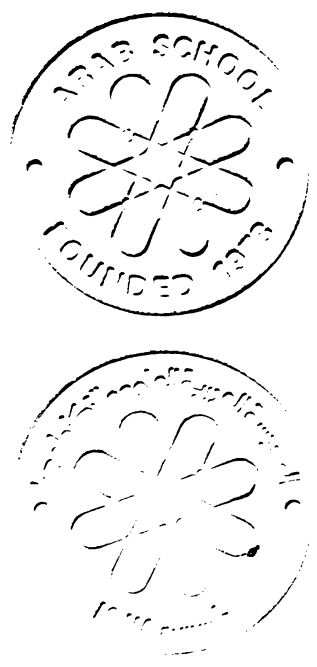
Computers and the Arabic Language



Edited by
Pierre A. MacKay

COMPUTERS AND THE ARABIC LANGUAGE

PROCEEDINGS OF THE ARAB SCHOOL OF SCIENCE AND TECHNOLOGY

J. Albaigés (editor)
Marine Pollution

R. Descout (editor)
Applied Arabic Linguistics and Information and Signal Processing

A. H. El-Abiad (editor)
Power Systems Analysis and Planning

T. Kailath (editor)
Modern Signal Processing

P. MacKay (editor)
Computers and the Arabic Language

P. D. T. O'Connor (editor)
Reliability Engineering

E. E. Pickett (editor)
A. Habal and F. Abosamra (co-editors)
Atmospheric Pollution

G. Warfield (editor)
Solar Electric Systems

FORTHCOMING

C. Foulard (editor)
Product Development and Production Engineering in Manufacturing Industries

R. Risebrough (editor)
Pollution and the Protection of Water Quality

COMPUTERS AND THE ARABIC LANGUAGE

Edited by
PIERRE MACKAY
University of Washington
Seattle, Washington, USA

HEMISPHERE PUBLISHING CORPORATION
A member of the Taylor & Francis Group

New York Washington Philadelphia London

COMPUTERS AND THE ARABIC LANGUAGE

Copyright © 1990 by Hemisphere Publishing Corporation. All rights reserved. Printed in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the publisher.

Cover design by Sharon DePass.

A CIP catalogue record for this book is available from the British Library.

1 2 3 4 5 6 7 8 9 0 BRBR 8 7 6 5 4 3 2 1 0 9

3C 2C

Library of Congress Cataloging-in-Publication Data

Computers and the Arabic language / edited by Pierre MacKay.
p. cm.—(Proceedings of the Arab School of Science and Technology)
Based on papers from a Summer Session of the Arab School of Science and Technology, held near Damascus, July 1985.

1. Arabic language—Data processing—Congresses. 2. Computational linguistics—Congresses. 3. Microcomputers—Programming—Congresses. I. MacKay, Pierre A., date. II. Arab School of Science and Technology. Session (1985 : Damascus, Syria)
III. Series.
PJ6074.C67 1990
492'.7'0285—dc20 89-39888
CIP

ISBN 0-89116-563-0

Contents

Contributors vii

Preface ix

The Arab School of Science and Technology xi

IMPACT OF COMPUTERS ON THE DEVELOPMENT OF THE THIRD WORLD 3
R. P. JHUNJHUNUWALA

INTEGRATION OF INDIAN LANGUAGES IN COMPUTERS AND COMMUNICATION SYSTEMS 7
R. P. JHUNJHUNUWALA

TRENDS IN MODERN COMPUTER–BASED SYSTEMS **17**

KNOWLEDGE-BASED AND EXPERT SYSTEMS IN UNDERSTANDING PROBLEMS 19
J.-P. HATON

OFFICE SYSTEMS AND INFORMATION SYSTEMS 29
C. A. ELLIS

ARABIC INPUT/OUTPUT TO COMPUTERS **53**

A SURVEY OF BILINGUAL PERIPHERALS 55
N. HARFOUCH and S. KOTOB

FONT DESIGN FOR COMPUTER-BASED DOCUMENTATION SYSTEMS 69
P. A. MACKAY

DESIGN CRITERIA OF ARABIC PERIPHERALS 78
M. MRAYATI and B. MOUNAJED

DESIGN OF ARABIC KEYBOARD LAYOUT BASED ON STATISTICAL PROPERTIES OF ARABIC CHARACTERISTICS 97
N. IDLEBI and M. MRAYATI

STANDARDS FOR ARABIC CHARACTERS IN INFORMATICS 102
M. A. AL-SALEH

THE ARABIC CODING STANDARD: AFTERTHOUGHTS AND DESIDERATA 107
P. A. MACKAY

ARABIC CHARACTER RECOGNITION 113
F. HAJ-HASSAN

ELECTRONIC SPEECH SYNTHESIS ARABIC COMPUTER SPEECH OUTPUT 119
M. MRAYATI

AUTOMATIC SPEECH RECOGNITION: STATE OF THE ART AND APPLICATIONS TO ARABIC 138
J.-P. HATON

ARABIC SOFTWARE **149**

ARABIZATION OF SOFTWARE TOOLS 151
A. S. ABU-MOSTAFA and S. A. ARAFEH

ARABIC ON NETWORKS AND MAIL SYSTEMS 169
P. A. MACKAY

GENERATION OF HIGH QUALITY ARABIC COMPUTER OUTPUT 171
S. SAMI and O. ALAMEDDINE

APPLIED ARABIC LINGUISTICS FOR INFORMATICS **183**

IMPACT OF LINGUISTICS ON INFORMATICS 185
S. ATTASI

STATISTICAL STUDIES IN ARABIC LINGUISTIC 190
M. MRAYATI

MORPHOLOGY AND SYNTAX OF THE ARABIC LANGUAGE 201
Y. HLAL

A COMPREHENSIVE ARABIC MORPHOLOGICAL ANALYSER GENERATOR 208
B. THALOUTH and A. AL-DANNAN

FROM A FREE ARABIC TEXT TO ITS WORD FREQUENCY TABLE: THE SEARCH FOR A SOLUTION 218
A. M. ABDELKARIM

BILINGUAL HARDWARE AND SOFTWARE: CASE STUDIES 223

CASE STUDY ON THE DESIGN OF BILINGUAL HARDWARE AND SOFTWARE 225
J. KARLSON

METAL: AN OPERATIVE MACHINE TRANSLATION SYSTEM 234
T. SCHNEIDER

INDEX 239

Contributors

Abdel-Karim, Abdin
Khartoum International Institute for Arabic
P.O. Box 26, Diume
Khartoum, Sudan

Abu-Mostafa, A. S.
Arabic Software & Computers, Inc.
725 Deep Valley Dr.
Rolling Hills Estates, California 90274, USA

Alameddine, O.
IBM, Kuwait Scientific Center (KSC)
P.O. Box 4175
Safat, Kuwait

Al-Dannan, Abdulla
Kuwait University
College of Education
Kuwait, Kuwait

Arafeh, Samir
ASCI
P.O. Box 2798
Palos Verdes 90279, USA

Al-Saleh, Mohamed Amin
SASMO
P.O. Box 281
Damascus, Syria

Attasi, Samer
IBM, Kuwait Scientific Center
P.O. Box 4175
Safat, Kuwait

Ellis, C. A.
Stanford University and Xerox Corporation
Stanford, California, USA

Haj-Hassan, Firyal
SSRC
P.O. Box 4470
Damascus, Syria

Harfouch, Nabil
SSRC
P.O. Box 4470
Damascus, Syria

Haton, Jean-Paul
University of Nancy
CRIN P.O. Box 239
54506 Vandoeuvre
France

Hlal, Yahya
Mohammed V University
P.O. Box 765
Agdal, Rabat
Morocco

Idlebi, Nibal
HIAST
P.O. Box 7028
Damascus, Syria

Jhunujhunuwala, Ramesh
CMC Ltd.
115 Sarojinideri Road
Secunderabad 50003
India

Karlson, J.
Siemens AG
Hofmannstr. 51
8000 Munchen 70, FDR

Kotob, S.
KISR
P.O. Box 24885
Safat, Kuwait

MacKay, Pierre
University of Washington
FR-35
Seattle, WA 98195, USA

Mounajed, Bachir
SSRC
P.O. Box 4470
Damascus, Syria

Mrayati, Mohamad
SSRC
P.O. Box 4470
Damascus, Syria

Sami, S.
IBM, Kuwait Scientific Center (KSC)
P.O. Box 4175
Safat, Kuwait

Schneider, Thomas
Siemens Ag
Hofmannstr. 51
8000 Munchen 70, FDR

Thalouth, B.
IBM, Kuwait Scientific Center (KSC)
P.O. Box 4175
Safat, Kuwait

Preface

The language of mathematics and formal logic is universal, and so long as computers were largely reserved for tasks that could be expressed mathematically, the problem of natural language interfaces received rather limited attention. This situation was already changing when the development of the small personal computer made it imperative to look seriously at the way in which nonspecialists might interact with computers. But only ten years ago, the solution often proposed to the sort of problem that is presented by the entry and display of Arabic language text was to alter the language, rather than to improve the computer.

The present volume is devoted to the exploration of all aspects of natural language interfaces with the computer. Not only is the visible text under consideration, but also speech recognition and speech synthesis. The focus is, of course, the Arabic language, but the papers in this study will be of interest to all who are concerned with natural language interfaces. The use of a particular pattern of morphology and syntax, as well as the development of a specific mode of writing, may result in a more difficult programming challenge in one phase of natural language processing, but offer a compensating advantage in another. Arabic, for example, is obviously more difficult to produce in satisfactory form on a CRT terminal than English, and also harder to print in hard copy. But the papers on speech recognition and synthesis suggest that the distinctive character of Arabic as a member of the Semitic language family may make the association of coded data-bases with speech input and output a good deal easier than in the case of English.

This volume arises out of a Summer Session of the Arab School of Science and Technology, held near Damascus in July 1985. The purpose of these Summer Sessions is to bring together experts in technology and advanced students from all over the Arabic world, in association with representatives from other countries specializing in related technological fields. The sessions are carefully balanced between theoretical and practical studies. One of the primary concerns of the Arab School of Science and Technology is to accelerate the progress of automation and computerization throughout the Arab World wherever it is appropriate, using appropriate levels of technology. There is therefore great interest in the critical evaluation of recent designs of hardware and software for the Arabic language. The results of these evaluations will provide guidance for future designs of systems that must offer a truly bilingual environment for computing.

That environment does not yet exist. Small parts of it have been developed, but the larger effort not just in the Arabization, but in a broader sense in the internationalization of hardware and software will take many years. One of the most difficult problems will be the creation of software systems that can be set to communicate with many different users, each in his or her natural language. The logical functions of the software must remain stable and unchanged, while the help and error messages change from user to user. The discussions at this Summer Session of the Arab School of Technology brought this problem out into the open, and the papers in this volume give a sense of the effort both inside and outside the Arab world to lay the necessary foundations for truly international systems of software.

Pierre MacKay

Arab School of Science and Technology

HISTORY

The Arab School of Science and Technology, a pan-Arab, non-profit organization headquartered in Damascus, Syria, was founded in 1978 by the initiative of the Kuwait Institute for Scientific Research (KISR), the Scientific Studies and Research Center (SSRC), and the Supreme Council of Sciences (SCS) in Syria to provide a high level continuing education program to Arab scientists in fields that are judged crucial to the development of the Arab countries.

Since its establishment the School has dealt with four major topics: electronics, energy, environment, and informatics. Additionally, two sessions were held in fields related to applied mathematics. The School has plans to expand into other areas of specialization, such as the transfer, adaptation, and development of technology in the Arab World.

The School has attempted to create for each topic a regular forum of scientific exchange in chosen areas of the Arab World. It has succeeded in establishing one for electronics in Syria, and another one for energy in Kuwait. The forums for the environment series and the informatics series were originally planned to be established in one of the Maghreb countries (i.e. Tunisia, Algeria, or Morocco), but until plans are finalized, the sessions in this series will be held in Syria.

In addition to its original main sponsors (KISR and SSRC), the School has also been sponsored by a number of national, regional, and international organizations which include Kuwait University, KIFAS, CNCPRST, ALECSO, UNESCO, UNDP, UNIDO, and IFIP as well as other such organizations.

OBJECTIVES

Through its program, the Arab School aims to fulfill the following objectives:

1. Familiarize Arab scientists, engineers, and university professors with the latest advances in science and technology through intensive, advanced post-graduate and highly specialized post-doctoral courses given by leading scientists.

2. Facilitate direct contact between Arab scientists to establish a propitious atmosphere for joint cooperation in the field of science and technology.

3. Encourage Arab scientists working abroad to return to their countries by providing them with opportunities to contribute to the School's activities and closely review the scientific resources of their countries.

4. Facilitate scientific cooperation among Arab and Muslim countries by direct contacts provided by the school.

5. Provide an overview of scientific activities in Arab countries by publishing session proceedings which cover scientific and technological developments in the Arab/Muslim world.

ARAB SCHOOL AUTHORIZED COMMITTEES

SCIENTIFIC PROGRAM COMMITTEE

Dr. M. Mrayati
Scientific Studies and Research Center
P.O. Box 4470
Damascus, Syria

Dr. B. Mounajed
University of Damascus
Damascus, Syria

Dr. N. Harfouch
Arab School of Science and Technology
P.O. Box 7028
Damascus, Syria

Dr. M. Farah
Higher Institute for Applied Science and Technology
P.O. Box 7028
Damascus, Syria

ORGANIZING COMMITTEE

Ms. M. Stephan
Administrator, International Relations
Arab School of Science and Technology

Mrs. H. Nahas Armanazi
Session Coordinator
Arab School of Science and Technology

Ms. A. Kouatly
Administrator, Local Arrangements
Arab School of Science and Technology

PREVIOUS SESSIONS	DATE		LOCATION
Solid State Electronics	Summer	1978	Syria
Communications	Summer	1979	Syria
Minicomputers, Microprocessors, and Their Applications	Summer	1980	Syria
Power Systems: Analysis and Planning	Winter	1981	Kuwait
Control Systems: Theory and Applications	Summer	1981	Syria
Solar Electricity Systems	Winter	1982	Kuwait
Pollution and Protection of the Water Quality	Summer	1982	Syria
Modern Signal Processing	Summer	1983	Syria
Applied Arabic Linguistics and Signal & Information Processing	Fall	1983	Morocco
Technology Transfer and Adaptation in the Arab World	Fall	1983	Syria
Informatics and Applied Arabic Linguistics	Summer	1985	Syria
Atmospheric Pollution	Summer	1985	Syria
Mathematical Modelling and Applications	Spring	1986	Syria
Reliability Engineering	Summer	1986	Syria
Marine Pollution	Summer	1987	Syria
Product Development and Production Engineering	Summer	1988	Syria
Queueing Theory and Applications	Winter	1988	Syria
Information and Computer Networks in the Arab States	Summer	1989	Syria

FORTHCOMING SESSIONS	DATE		LOCATION
Application of Remote Sensing to Hydrology and Water Resources	March	1990	Syria
Planning in the Scientific Research and Engineering Development Sectors	July	1990	Syria
Microelectronics (conference)		1990	Syria
Data Bases and Their Arabization		1990	Syria
Computer Aided Design		1991	
Composite Materials, Their Technology and Application		1991	
Sensor		1992	

COMPUTERS AND THE ARABIC LANGUAGE

Impact of Computers on the Development of the Third World

R. P. JHUNJHUNUWALA
CMC Limited

1.0 INTRODUCTION

Computer-based information and control technology has been the key to the evolutionary advances in the industrial, technological and socio-economic developments in the world for the last 30 years. Four Generations of computer technology have been developed and deployed in this short span and the advanced countries have launched massive R&D programmes to develop the Fifth Generation technology for computers to meet their needs for the 90's and the 21st century in the social, industrial and defence sectors. While this technology is permeating in all walks of life in the developed world, the third world has to catchup and exploit the computer technology in a meaningful manner for their development programmes. There is an urgent need for promotion, rapid introduction and effective utilization of computer technology in the vital sectors of national growth and for improving the quality of life in the developing countries. Policies, plans and organizations for computerization have to be developed hand-in-hand with their vital sectorial programmes for development. In order to discuss the impact of computers on the development of the third world, this paper examines :

i) real needs of these countries for development oriented computer-based applications,

ii) nature of the computer-based systems required for such applications, and

iii) the significant factors that affect the introduction and utilization of this technology in these areas.

It is emphasized that indigenous Systems Engineering and support infrastructure are essential in order to deploy this technology in any meaningful manner. This paper is prepared based on the experiences of CMC Limited, A Government of India Enterprise, in its attempts to establish an indigenous infrastructure for providing total computer support services with an emphasis on meeting the needs of vital sectors of national development. It also draws upon the experience derived from a United Nations supported project - Project INTERACT, which was undertaken by Government of India on the basis of Technical Cooperation among Developing Countries (TCDC) for development of knowhow in systems engineering of Online Realtime computer-based systems dedicated to development oriented applications in Electric Power, Railways and Meteorology.

2.0 REAL NEEDS FOR COMPUTERS

For having any meaningful impact of computers in the national development process, computers need to contribute i) in improving productivity and efficiency in the industrial and service sectors; ii) support the scientific and technological progress and iii) help improving the quality of life. For various development programmes, computers need to be deployed in the socio-economic sectors such as

- agriculture, mining, fisheries, forestry and other primary production areas;
- manufacturing industry;
- service sector such as communication, transportation, water supply etc.;
- generation and distribution of Energy such as Electric Power, Oil, etc.;
- community services such as Education, Health, publishing, newshandling;
- government administration and management services; and
- engineering, scientific and research areas.

Levels of usage of computers in these application areas largely depend on the level of knowhow available in the country for application engineering; type and level of infrastructure available such as data communication facilities, maintenance, training, enhancement support, etc. In order to understand the requirements for deployment of computers in these areas, it is necessary to analyse the nature of possible applications and capabilities/facilities required for computerizing these applications.

3.0 NATURE OF COMPUTER APPLICATIONS

For the purpose of this discussion, the nature of the applications in most of these relevant sectors can be categorised broadly into the following three groups:

i) Data Processing
ii) Management Information and Decision Support Applications
iii) Specialized applications.

The capabilities and facilities required for each of these broad application categories are now analysed to understand their deployment for development purposes.

3.1 Data Processing

These type of applications involve tabulation, indexing, classifying and sorting with storage and retrieval operations for large volume of data. These applications are used to mechanize clerical operations for example, in commercial establishments for bookkeeping, invoicing, sales accounting, payroll, inventory accounting, etc. Most of the initial applications of computers involved this type of processing and a major usage of this type is prevalent in many of the third world countries. These use general purpose machines and languages like COBOL/BASIC and do not involve high level of expertise in application development and implementation. Such application systems are maintained by the end-users and involving little support from outside. Even though these applications are prevalent in many of the developing countries, their impact on developmental activities is not significant as these applications are mostly labour saving and hardly mature into more significant management applications.

3.2 Management Information and Decision Support Applications

These type of applications are required for general administration and management of industry, government and community services. Their contribution is higher for the growth as compared to DP applications. These applications involve information storage in a data base, query and retrieval; PERT/CPM, production planning, OR and simulation methods, etc. These applications normally run on general purpose machines with on-line and data base facilities. Standard packages supplied by vendors could be adapted with some customization effort. These systems require higher skills in system analysis, design and maintenance support as compared to data processing type of applications. The computerization of these applications largely depend on availability of technical skills, training facilities, technical features and functions of the marketed systems in the country.

3.3 Specialized Applications

In this category, the computer-based systems facilitate the major functions of the organization. Some applications in this category could operate in off-line mode while major applications call for on-line operations. The on-line system operate in dedicated mode becoming an integral part of the process/operation. Examples of dedicated systems are:

- Airline/railway reservation systems
- Freight operations management systems
- Process monitoring and control in manufacturing industry
- Monitoring and control systems for generation and distribution of Electric Power
- On-line systems for banking
- Communication systems for switching of voice, messages, etc.
- Office support systems
- Cartography system
- Computer aided design
- Computer aided instruction and so on.

Such systems require many advanced technical facilities such as on-line transaction processing, real time processing, data communications, graphics & image processing, text processing and natural languages, modelling, etc. Such systems normally require special firmware, customized software and consist of system elements such as communication, computer, remote data collection and control terminals and specialized man-machine interface. Such applications would normally require high availability and a reliable data communication infrastructure for their meaningful impact. High level skills are required in computers, communication. specialized firmware, system analysis & design, and system support for enhancements and replacements. A sound maintenance infrastructure is necessary as the operations of the organization depend on the functioning of such dedicated systems. This category also provides for other specialized applications not included previously and to cover scientific, engineering and research areas. Design and development of such systems can be taken up by specialized groups with good knowledge in computerization and expertise in specialized applications. These type of application systems are necessary for the advancement of scientific and technological base.

Such specialized application systems make a real impact on the national development oriented activities. However, it needs to be emphasised that to introduce and operate such systems, an advanced level of expertise and support infrastructure are essential.

4.0 FACTORS AFFECTING COMPUTERIZATION FOR DEVELOPMENT IN THE THIRD WORLD

Based on the real needs and the nature of applications for computerization as discussed above, it is apparent that the level of contributions go higher when the computerization level moves from data processing to management applications and to specialized applications. The level of knowhow and technical support required also rises with the level of computerization.

In order to move to higher levels of computerization in management and specialized application areas, it is necessary to review the current scenario in the developing countries regarding supplies, deployment and support of computer-based systems :

a) To a very large extent the computer-based systems are imported involving scarce foreign resources.

b) The computerization has been primarily promoted and supported by the multinational vendors or their agents keeping in view, naturally, their business interest.

c) Data processing applications are promoted initially in most of the third world countries due to larger numbers involved in supplies of standard general purpose machines and less efforts/investments required for market creation, training and support. Many of the third world countries have not progressed, beyond data processing and limited management type of applications, due to total dependence of these countries on the few foreign vendors who got established themselves for supplies and support in these countries.

d) Many of the computer-based systems for dedicated applications are procured with international financial loan arrangement or under some aid programme in many third world countries. These systems are procured as per the procedures of these financing institutions and are mostly supplied by multinationals from the industrialized world on turnkey basis. Attempts are made to transplant the applications as operational in the developed world without adequate preparations and often these do not directly match the requirements or operational practices in the third world. The knowhow also does not get transferred for

effective utilization and subsequent support for modifications, enhancements and maintenance. Thus, the turnkey option does not give effective solution from the long term point of view.

e) Many developing countries do not have adequate reliable communication infrastructure for supporting the data communication which is essential element of most of the real-time/on-line application systems. A long gestation period and large investments are involved in providing and supporting such infrastructure. Computerization of many meaningful applications could thus take place along with the building up of the necessary communication infrastructure for various development oriented activities as per the national priorities.

5.0 NEED FOR INDIGENOUS TECHNICAL INFRASTRUCTURE

The current status on supplies, deployment and support of computer-based systems indicate that computerization for meeting the real needs could not be achieved in many third world countries due to inadequate level of knowhow and technical infrastructure available for such purpose. As the application level from routine commercial data processing moves to management and further to dedicated specialized application systems it is imperative to have i) a system support infrastructure for providing hardware and software maintenance, modifications, enhancements, upgradation and replacements; ii) systems engineering skills for system analysis and design; system hardware selection & integration; specialized firmware and software adaption/development; and system implementation.

5.1 Indigenous Support Infrastructure

In order to effectively utilize this technology for higher level of applications, the system support infrastructure has to be available indigenously as vital sectors of national operations would totally depend on the functioning of the computer-based systems. Such an indigenous support infrastructure has been established in India and its effects are worth discussing here. Government of India had set up an organization - CMC Limited in 1976 to provide one point support services in hardware and software maintenance; installation & commissioning; systems engineering; software development; computer centre services; education & training; etc. The major benefits due to establishment of such a support infrastructure have been manyfold as discussed now.

Prior to CMC it was normally possible to procure only those computer systems for which the support services were available from the vendors within the country. Since there were only two major multinationals who were providing maintenance support, the systems promoted and offered by them were mostly procured due to essentiality of such support. Since establishment of CMC, now in India there are systems supplied by as many as 35 vendors as support services are ensured for the country by CMC. Thus appropriate system could be procured as per the needs of the applications and also multiple options are made available for selection. Nature of applications have also advanced from routine commercial data processing to production planning & control, MIS for industries and government administration, on-line reservation systems, realtime monitoring and control systems, computers for schools, systems to support scientific and research purposes, etc. Since it is an organization to provide support at national level, it has been also possible to optimize resources such as spare parts, specialized tools & test equipment, specialized manpower, etc. The foreign exchange outgoing for such services has also been reduced due to indigenous services. Problems related to foreign exchange are being experienced by many countries due to tie up of maintenance/rental charges of computer equipment with the US dollar by the multinational corporations. With the depleting value of local currency, for example, these charges shot up almost 10 times in a matter of days in some Latin American countries not long time ago. Such a hike, suddenly, made the computerization as financially non-viable. Problem is also faced by some countries due to stoppage of support services as the multinational supplier suddenly decided to close their operations. Such a event happened in India and fortunately CMC existed to take over support operations of over 900 installations. It has also been possible to provide support for extended periods, for example, some installations are as long as 20 years old. The Indian experience indicates that the indigenous management and technical expertise infrastructure are essential elements of any viable computerization programme.

5.2 Indigenous Systems Engineering Capabilities

As the nature of computer usage moves from data processing to dedicated and specialized applications, largerefforts and expertise are required in system analysis & design; customizing or developing the application system; training; and support for operations. As discussed earlier, vendors from the developed world may not promote such specialized applications unless there is a sizable market, as this involves efforts for creating a market, specialized support from skilled manpower and long gestation periods. It is also found that the transplanted systems procured on turnkey basis have been ineffective due to lack of proper systems engineering of such systems as per the real needs of the end-user and lack of transfer of knowhow to provide effective support for such systems. Hence, it is necessary to establish systems engineering capabilities indigenously in the developing countries for introducing meaningful applications. It is feasible to set up such capabilities as it involves activities for system analysis and design; selection of system elements such as computers, communication equipment, terminal units, etc.; application software development; system integration and operations; etc. which are manpower oriented. Such technical manpower is available in many countries and also it is possible to provide training in case such manpower is not available. Such indigenous systems engineering capabilities would enable to save valuable foreign exchange, complement indigenous manufacturing and provide long term support facilities for application of computers for development of infrastructure in the third world.

Keeping in view the need for systems engineering, knowhow has been developed in India in systems engineering. Some of the major on-line realtime application systems which have been developed by CMC are for passenger reservation system for railways; message switching system for civil aviation, news agencies and meteorology; computer-based learning; Indian language based systems; etc.

In this regard it would be worthwhile to present experiences on developing knowhow in systems engineering under a project - Project INTERACT undertaken by the Government of India with support from United

Nations agencies. Under this project, knowhow was developed for systems engineering computer-based systems in the national priority sectors of Power Systems Management, Railway Freight Operations Management and for Meteorological applications. The project was implemented with the participation of professionals from developing countries on the basis of Technical Cooperation among Developing Countries (TCDC). 23 developing countries from Latin America, Eastern Europe, Africa and South East Asia participated in the technology development and training activities of the project. About 200 professionals including management personnel were trained from these countries and India in the systems engineering techniques for developing real time on-line computer-based systems for monitoring and controlling generation and distribution of power; for railway freight operations management system and for image processing of satellite picture for meteorological applications. Sixteen experts from developing countries participated in the technology development activities.

The exposure to developing world during Project INTERACT have indicated that :

i) There is a need to create awareness in government administration and management about the significant role computers can play in national development oriented activities. An awareness that information is a resource has to be brought about at all decision-making levels for making any headaway towards computerization. Also there is a need to promote system engineering approach for building up capabilities to effectively utilize and support this technology.

ii) There is a large need for education & training in computers and related topics. There was a very high demand for training courses offered under 'INTERACT' for computers and their sector oriented applications.

iii) The level of computerization in most participating countries was limited to data processing and some management oriented applications due to the reasons indicated earlier in this paper. Wherever dedicated application systems in the specified sectors were installed they were obtained through imports on turnkey basis under some financial assistance programme. The recipients had difficulties in modifying, enhancing and supporting such systems as it was indicated that the training provided by suppliers was inadequate and there was virtually no knowhow transfer.

6.0 CONCLUSIONS

Keeping in view the experiences of CMC and Project INTERACT and in order to have any meaningful impact of computers for the development of the third world, it is necessary

i) to improve consciousness that information is a resource

ii) to ensure indigenous support infrastructure and

iii) to set up systems engineering capabilities.

As computer would be playing a very significant role in the industrial and socio-economic activities of the third world hopefully in not far away future, and keeping in view that embargoes on supply of technology and products from the developed world are being imposed, the third world would also have to consider seriously the development and production of this technology indigenously.

Integration of Indian Languages in Computers and Communication Systems

R. P. JHUNJHUNUWALA
CMC Limited

1. INTRODUCTION

The use of Indian languages in Government transactions and education has gained importance since Independence. This has spread to telecommunications resulting in the development of Devanagari Teleprinter in 1968 [1]. For over two decades, many national institutions and research workers have been engaged in providing input/output and processing of Indian languages in computer systems [2]. Initially, the complexity of Indian scripts posed serious problems in the development of practical devices. However, with the availability of low cost microprocessors and large scale integrated circuits, practical schemes and affordable devices started emerging in the past five years.

In an independent application of far reaching effects namely the Printing and Publishing Industry, the use of Indian languages has a much longer history. Indian scripts have been offered through various technologies ranging from handsetting through hotmetal and photocomposition to electronic composition based on CRT and laser technologies. Most of these typesetting systems have originally been developed for English or Roman based scripts and adapted for Indian scripts [3, 4].

To asses the status of development and accelerate the implementation of Indian scripts in computers, telecommunications and publishing, the Department of Electronics of Government of India had organised an international seminar on the subject in November 1978. The Department of Electronics and the Ministry of Communications had setup a committee to establish national standards for coding and keyboarding of Indian scripts - INDIAN SCRIPT STANDARD CODE FOR INFORMATION INTERCHANGE ISSCII-8 1 This standard offers an 8 bit code for a superset of all Indian languages of Brahmi family based on the common phonetic ordering of the basic alphabet. The standard also prescribes a phonetic based keyboard layout for enhanced Devanagari (the script of Hindi, Marathi and Sanskrit.).

CMC Limited which had proposed and pursued a unified approach for coding and keyboarding for all Indian languages has taken an active part in the evolution of the ISSCII-8 standard and developed a range of products to be based in computer systems (input/output devices) wordprocessing, telecommunications and typesetting. This paper describes the features of Indian scripts and the evolution of national standard ISSCII-8 followed by a presentation of CMC's LIPI-PRODUCT LINE for Indian Languages (the material is drawn from earlier publications of CMC and the standards reports) [1,4,5].

2. FEATURES OF INDIAN SCRIPTS

Indian scripts are characterised by a large-set of letter patterns. The basic alphabet consists of 16 vowels, 36 consonants and a few diacritical marks. Two typical scripts, namely, Devanagari and Telugu are shown together in Chart 1 to demonstrate the commonality of phonetic ordering and the sharp contrast of graphic representation [4,5].

In Indian languages, the syllables are generated from an ordered sequence of consonant(s) to a vowel based on the phonetic principles. In the script, such a syllable is presented as an integrated graphic pattern which is called a composite character (variously termed as Samuktakshar or conjunct or ligature).

A composite character is formed by taking the primary symbol of one of the constituent consonants of the syllable and appending to it the secondary signs of the other consonant(s) and a vowel. Certain consonants and vowels have multiple secondary forms as shown in Chart 2. In cases where no vowel is associated with a consonant or a combination of consonants, a special symbol (Hallant in Devanagari and Pollu in Telugu) is appended to form a composite letter. At the time of forming a composite character, both the primary and secondary patterns undergo modifications. These modifications are directed by the convenience of writing, typing or typesetting. As a result, a standardised procedure of forming composite letter patterns does not exist. A few examples of Devanagari and Telugu composite character formation is given in Chart 3.

3. PHONETIC BASED REPRESENTATION

Although the formation of composite patterns is ad-hoc, there is an excellent and consistent method of representing composite characters as an ordered sequence of the constituent consonant(s), vowel and diacritical mark (Phonetic elements). This method which is based on phonetic principle is consistent within a script and across the scripts of Brahmi family. Examples of representing the words COMPUTER and RASHTRIYA in Devanagari and Telugu scripts are given in Chart 4A.

While it is possible to represent words as a sequence of phonetic elements, it is necessary to group the phonetic elements of a syllable that combine to form a composite character. If the grouping information is not preserved, the sequence of phonetic elements is liable to be interpreted differently as shown in Chart 4B.

4. PHONETIC BASED CODING

Coding is a method of representing text in machine readable form for the purposes of storage, processing and transmission. Indian language text can be represented unambiguously as an ordered sequence of phonetic elements if grouping information is also indicated in the sequence.

The Standards Committee has therefore introduced a LINK symbol to represent the grouping of phonetic elements that combine to form a composite character. Thus, the words COMPUTER and RASHTRIYA can be represented as an ordered sequence of phonetic elements and LINK. (Chart 5A) Since the diacritical marks implicitly attach themselves to the previous characters, the LINK need not precede the diacritical marks.

The statistical analysis of Indian language texts revealed that vowels combine with previous consonant(s) and appear in their secondary forms (Matras) with a frequency of 30% to 35%. In contrast, the stand-alone vowels in their primary form occur with a frequency of only 3 to 4%. Thus, it is economical for storage and transmission time to assign a single 8-bit code to the sequence of LINK and VOWEL (See Chart 5B). Accordingly, the code chart for ISSCII-8 (See Chart 6) accommodates stand-alone vowels, consonants and secondary vowel signs in that order. This arrangement yields collation in alphabetical sequence by sorting the codes in increasing numerical value [3,4,5].

5. PHONETIC BASED KEYBOARDS

In principle, any text can be keyed-in with a keyboard providing only consonants (35), vowels (16) and accent (DIACRITICAL) marks (5). Since the vowels and consonants may occur individually or in combination for composite characters, the keyboard must provide for indicating the grouping information also. A LINK key is introduced for this purpose. The accent marks implicitly attach to the previous consonants or vowels and so LINK need not precede them (See Chart 5B) [3,4,5].

5.1 Practical Keyboard for Minimising Keystrokes

The efficiency of keyboard design lies in minimising the number of keystrokes to enter text accurately and completely at a high speed.

As discussed earlier, if the keyboard consists of only consonants, vowels and accent marks, the LINK key must be operated 30% to 35% along with vowel key, to key-in the vowel signs. It is therefore, decided that vowel signs (Matras) must be provided as standard in the UNSHIFT positions of the keyboard. This requires 16 additional keys.

However, since analysis of text shows that the frequency of all the vowels (other than "A") is only 3% in Devanagari and 1.9% in Telugu, it was decided that stand-alone vowels will not be provided on the keyboard. Furthermore, stand-alone vowels do not normally occur in the middle of a word and it is possible to interprit a vowel sign as a stand-alone vowel by context (i.e., when it does not follow a consonant). In the exceptional cases (less than 1%) when a stand-alone vowel occurs after a consonant, LINK key has to preceed the vowel sign. Ironically LINK in this case acts as a DELINKING sign.

The keyboard of 43 to 46 main keys accommodates parivardhit (enhanced) Devanagari symbols, standard 'Samyuktakshars' and special accents, in addition to numerals, punctuation marks and mathematical signs. The control and function keys have to be added.

Based on the frequency analysis and occurrence of consonant-vowel statistics, the standards committee recommended a keyboard layout. This layout however, displaced punctuation marks (e.g. '.' and ',') and mathematical signs ('+' and '=') as compared to the ANSI standard keyboard. Because of the bilingual requirement, CMC's implementation keeps the punctuation/mathematical signs as per ANSI standard (See Chart 7).

Our experiments on phonetic based keyboards and demonstrations of the keyboard have revealed their acceptance for use (particularly when the operator has no preconceived apprehensions). It is significant to note that the complications of the script forms do not effect the keyboarding. Thus the same keyboard can be used for teleprinters or typesetting without any degradation of quality or variety of output.

6. TECHNOLOGICAL CONTRIBUTIONS BY CMC

6.1 Indian Languages

Indian languages have a variety of character forms. Though the number of consonants, vowels and accent (diacritical) marks is limited, the number of patterns that result as a combination of these, is extremely large. (Chart 2 shows a few examples). For computerisation, storage of all these forms individually requires a huge memory. A practical solution is to code based on phonetic structure and synthesise composite character patterns on the output device.

6.2 Multilingual Word Processing

A multilingual word processor named - LIPI has been developed at CMC. LIPI is capable of simultaneously handling English and two Indian languages. One can mix these three languages at any point of text. Phonetic based keyboarding for Indian languages enables the use of single keyboard for all the languages. A single key depression changes the keyboard from Indian language to English or vice versa; and a control code switches it between the two Indian languages. Character display, printing and text formatting have been handled by specially designed hardware and software (proprietary).

6.3 Display Character Generation

While generating the display characters, following provisions have been made using hardwired logic, microprocessor and its software, and dot pattern data stored in ROMs.

a) Large matrix size is required because of the curves and loops and the occurrence of secondary signs above and below the main character. Devanagari and Telugu scripts need a body height of 10 dots and ascender height of 5 dots. The descender depth of 5 dots is sufficient for Devanagari, whereas Telugu requires 10 dots to include "vattu" signs which are as complicated as the main characters. A total height of 25 dots is required for Devanagari and Telugu. For letter quality printing about 35 dots may be needed.

b) Even the stand-alone characters are not of uniform width, let alone the complications due to secondary signs. Proportionate spacing is therefore a necessity for Indian scripts.

c) Since all possible character patterns cannot be stored individually, the character generator must have a capability to derive composite character patterns by merging elemental patterns stored to ROMs.

d) The cursor width must match the width of the character to which it is pointing. For this purpose an elastic-cursor has been provided.

Using these features, letters of very high quality could be displayed. The conventional letter forms and proper matra alignments could be attained using special design techniques, which were not possible by conventional typewriters (See Charts 9A and 9B). A special character generator printer circuit board in LIPI ensures these features.

6.4 Printer Driver

Good quality Indian language printing cannot be handled by Daisy wheel kind of printers due to the complex and varied letter forms. The required printers should provide for the points (a), (b) and (c) mentioned above. These features could be attained only by Dot Matrix Printers, with dot addressable graphic. The character generation is done by the host computer. In LIPI word processor, this is done by INTEL 8088 microprocessor and the script generating routines. Such a scheme enables using the same printer for all the Indian languages. We have used various dot matrix printers, like Philips GP300, IDS Microprism and Printronix B-600. The driver software, being written in higher level language like PASCAL and FORTRAN, enables portability of the software.

This software is also being used for printing Railway Reservation charts and Railway tickets, on VAX-11, under the Project "IMPRESS", being executed by CMC Limited.

6.5 Formatting

LIPI provides a versatile formatter, using which text can be aligned to left and/or right margin, it can be indented, centered, or even columnated, paragraphs may be indented and spaced out to the desired value. All standard formatting features are provided for a mix of English and two Indian languages.

In general, text in Indian languages is not hyphenated. Even in cases where it is done, the writing system of Indian languages does not permit a syllable to be broken (since syllables are represented as a single composite letter). Thus, hyphenation can be done by rules not requiring elaborate hyphenation dictionaries. In practical implementation we have not faced the necessity of hyphenation since word spaces could be uniformly adjusted in single dot increments (1/44 of an inch). The text thus formatted does not leave any uneven white space.

6.6 LIPI Product Range

Sections of LIPI-WP are being used to design other related systems, namely, a multilingual Video Terminal - LIPI-VT, data entry machine - LIPI-DE, teleprinter LIPI-TP. Also, we are enhancing the existing capabilities to the multilingual phototypesetting - LIPI-Comp.

On software side, we have successfully implemented transliteration of passenger names from English to Hindi for Indian Railways with approximately 80% accuracy.

REFERENCES

1. Report of the Committee set up to evolve design of Devanagari Keyboard for Electronic Teleprinter to be manufactured by Hindustan Teleprinters Limited, Madras, Ministry of Communication, Government of India, 1984.

2. Om Vikas, Use of Non-English Languages in Computers - A Selected Bibliography, Electronics Commission, Information, Planning & Analysis Group, Government of India, October, 1978.

3. Madur, S. P., Wakankar, L. S., and Ghosh, P. K., Design Information on Text Composition in Devanagari, Technical Report of National Centre for Software Development and Computing Techniques, Research Institute for Newspaper Development, 1980.

4. Narasimham, P.V.H.M.L., Narasimham, G.L., Ramakrishna Rao, G., Design Information on Text Composition in Telugu, CMC Limited and RIND, June, 1981.

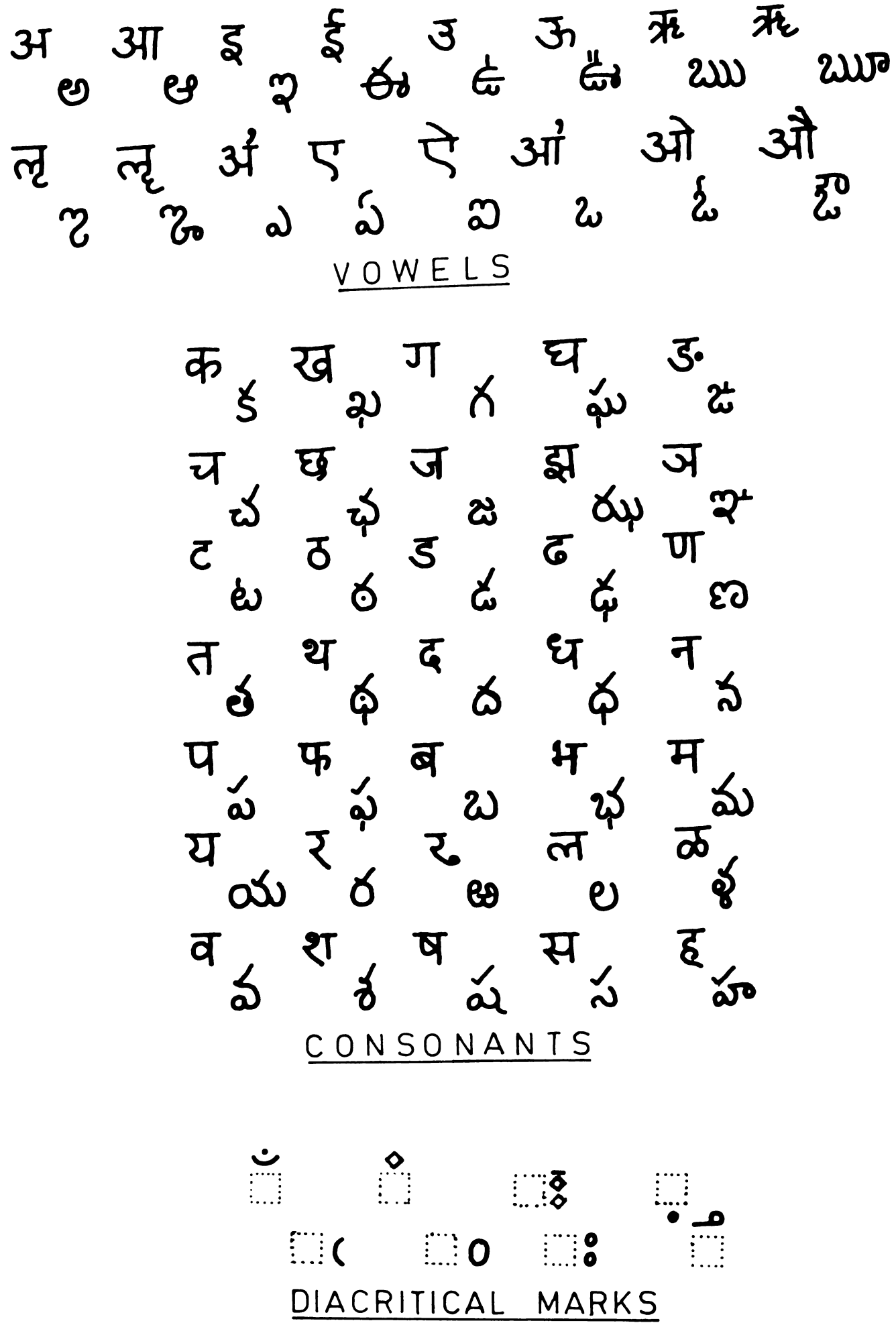


CHART-1: BASIC ALPHABET OF DEVANAGARI & TELUGU

NOTE : IN EACH PAIR OF ROWS THE FIRST ROW IS DEVANAGARI AND THE SECOND IS TELUGU.

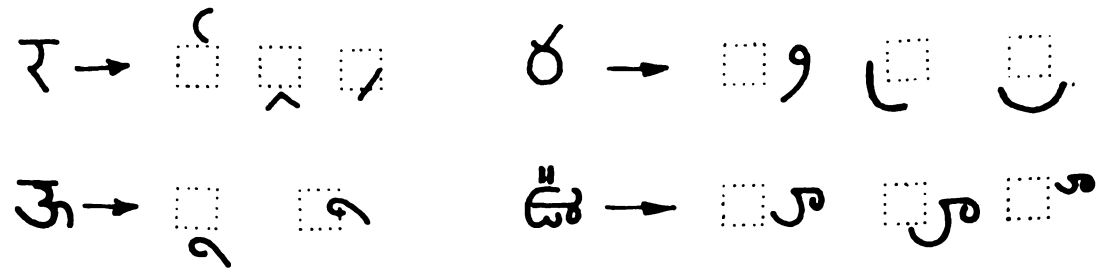
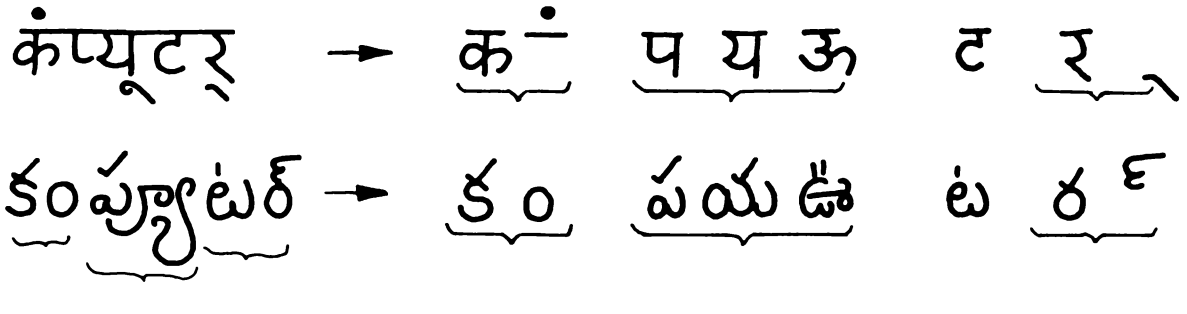


CHART-2 MULTIPLE SECONDARY FORMS

SYLLABLE	PHONETIC SEQUENCE	PRIMARY SECONDARY FORMS	COMPOSITE CHARACTER
PRE	प र ए	ट ◌ ◌े	प्रे
	ప ర ఏ	ఎ ◌ ◌ే	ప్రే
PSA	प स आ	ट स ा	प्सा
	ప స ఆ	ఎ ◌స ◌ా	ప్సా
VYU	व य ऊ	ठ य ◌ू	व्यू
	వ య ఊ	వ ◌్య ◌ూ	వ్యూ
SHTRI	ष ट र इ	ष ट ◌ृ ◌ि	ष्ट्रि
	ష ట ర ఇ	ఎ ◌ట ◌ి	ష్ట్రి
LKI	ल क इ	ल क ◌ि	ल्कि
	ల క ఇ	ల్ ◌క ◌ి	ల్కి

CHART-3 FORMATION OF COMPOSITE CHARACTERS



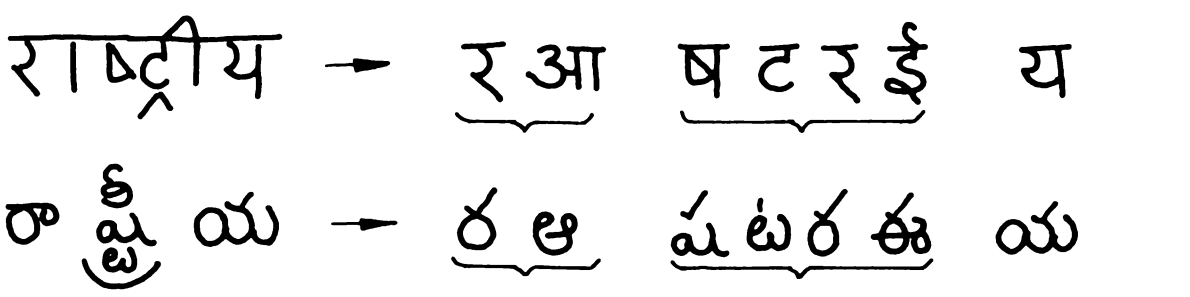


CHART-4A REPRESENTATION OF WORDS BY PHONETIC ELEMENTS

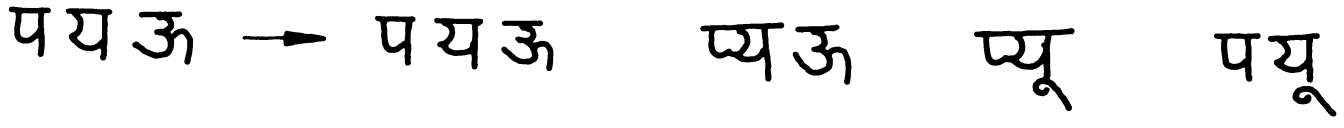


CHART-4B AMBIGUITY IF GROUPING IS NOT INDICATED

CHART- 4 PHONETIC BASED CODING AND GROUPING

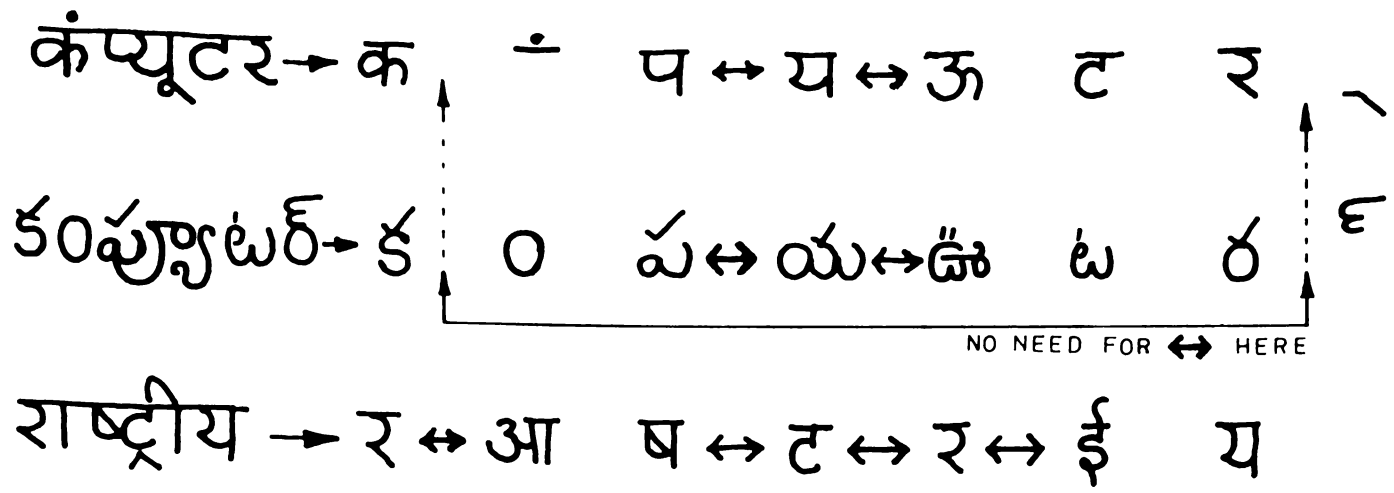


CHART-5A ENCODING WITH CONSONANS STAND ALONE VOWELS AND DIACRITICAL MARKS.

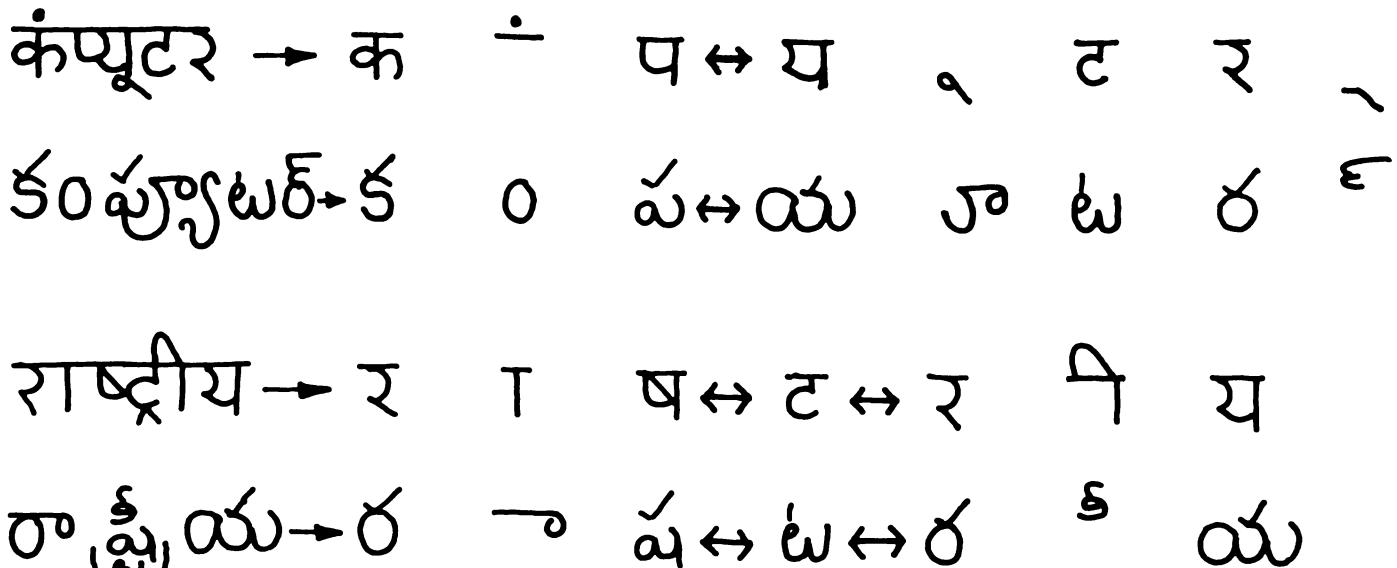


CHART-5B ENCODING WITH CODES OF VOWEL SIGNS.

CHART-5 GROUPING OF PHONETIC ELEMENTS THROUGH LINK

NOTE : NOTE THAT THE VOWEL SIGNS REPRESENT THE PAIR OF LINK AND THE CORRESPONDING VOWEL

	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	EXT	DC3	#	3	C	S	c	s
4	EOT	DC4	Rs	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EN	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	↑	n	~
F	SI	US	/	?	O	—	o	DEL

	0	1	2	3	4	5	6	7
0			ॐ	ऍ	ऋ	फ	़	---- HALANT
1			ँ	अं	ऌ	ब	ा	॑
2			ं	ए	ट	भ	ि	↔ LINK
3			ः	ऐ	ठ	म	ी	@ [4]
4			D-SP रिक्त	ऑ	ड	य	ु	रु [4]
5			ऽ AVAGRAH	ऑं	ढ़	र	ू	॒
6			अ	ओ	ढ	ऱ [2]	ृ	क्ष
7			आ	औ	ढ़	ल	ॄ	त्र
8			इ	क	ण	ळ	ॅ	श्र
9			ई	ख	त	ऴ [3]	ॆ	क्र
A			उ	ग	थ	व	े	
B			ऊ	घ	द	श	ै	
C			ऋ	ङ	ध	ष	ॉ	
D			ॠ	च	न	स	ॊ	
E			ऌ	छ	ऩ [1]	ह	ो	
F			ॡ	ज	प	ऺ	ौ	DEL

USER DEFINED SYMBOLS → SUGGESTED FOR DEVANAGARI

[SEPT 1983]

SAME AS ASCII-7
$ REPLACED BY Rs

1. ADDITIONAL IN TAMIL.
2. " IN KANNADA TAMIL MALAYALAM TELUGU.
3. " IN KANNADA TAMIL MALAYALAM.
4. TO BE REPLACED BY CORRESPONDING SYMBOLS IN OTHER LANGUAGES

CHART-6 INDIAN SCRIPT STANDARD CODE FOR INFORMATION INTERCHANGE (ISSCII-8)

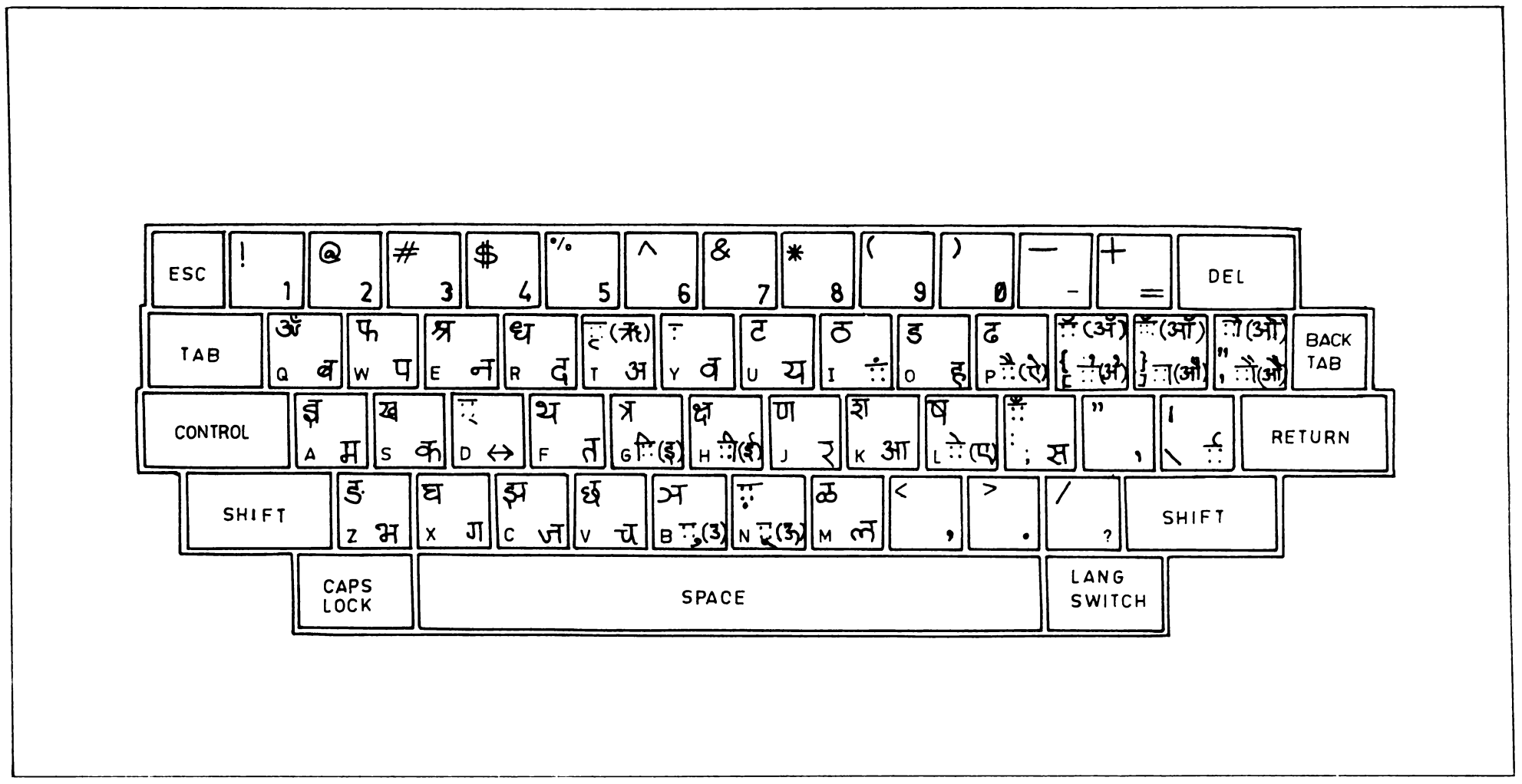


CHART-7 INDIAN LANGUAGE KEYBOARD

(INCLUDES STANDARD QWERTY LAYOUT)

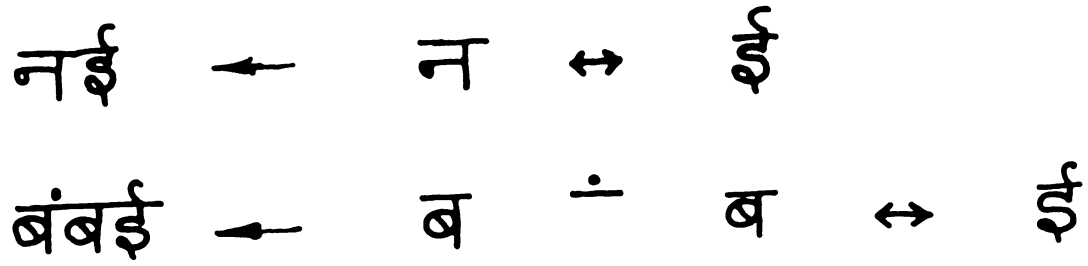


CHART-8 USE OF LINK KEY TO DERIVE STAND ALONE VOWELS

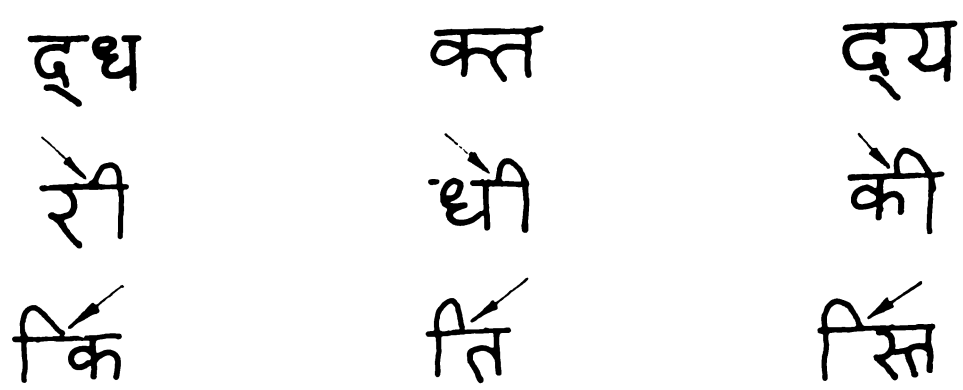


CHART-9A MODIFIED FORMS AS PRINTED ON TYPEWRITER

CHART-9B CORRECT FORMS AS PRINTED BY DOTMATRIX PRINTER

CHART-9 SPECIAL FEATURES OF INDIAN LANGUAGES

TRENDS IN MODERN COMPUTER–BASED SYSTEMS

Knowledge-Based and Expert Systems in Understanding Problems

JEAN-PAUL HATON
CRIN, INRIA
B.P. 239
54506 Vandoeuvre Cedex, France

ABSTRACT

Artificial Intelligence is now being widely used in industrial applications after several decades of research in the laboratories.

Knowledge-based techniques are mainly responsible for these new developments in a number of different domains. This chapter is an attempt at giving an overview of the field. We first give the basic principles underlying the design of expert systems. Then we present the impact of knowledge-based reasoning in understanding systems, with an emphasis to speech understanding and computer vision.

Basic principles are illustrated throughout the chapter by practical examples of operational systems.

1. INTRODUCTION

Artificial Intelligence (AI) has been experiencing for the past few years a rapid growth in research and development. Such areas as expert systems, natural language understanding, speech recognition, computer vision, etc. have significantly progressed, even though important problems still remain to be solved.

The present success of AI mainly results from the design of new system architectures which are able to use all the knowledge, including human expertise, available in a given domain. That is, these knowledge-based systems thus take human expertise into account in order to improve their own performances. In this chapter we will consider the application of knowledge-based techniques to understanding systems, especially speech understanding and computer vision where the basic problem consists of interpreting input physical data. In these two related fields there exists a close interaction between numerical data-processing (perceptual aspects such as in signal processing and pattern recognition techniques) and symbolic computation (cognitive aspects). Moreover, it is difficult to implement reasoning processes, due to the multiple knowledge sources and to the fact that data are incomplete and/or erroneous. The solutions which are found in these fields are therefore often original and powerful, and they are then used in other domains of AI.

After having presented the fundamentals of expert systems, we will concentrate on two major questions, i.e. the representation and use of knowledge and the structures for controlling efficient search strategies. The principles presented will be illustrated by practical examples from the fields of speech and vision.

2. FUNDAMENTALS OF EXPERT SYSTEMS

2.1. Basic Ideas

AI has achieved some of its most spectacular success in the field of expert systems. This area of research and development has concentrated on the design and implementation of sophisticated computer systems which are able to reach human expertise in very narrow, specialized domains of activity. These systems mainly rely on an explicit use of the knowledge that underlies human expertise. In that sense expert systems highly differ from early work in AI that aimed at developing general problem solving methods, independently of an application domain (Newell 1963) : efficiency has been sought for at the expense of generality.

A fundamental idea that emerged from the very early expert systems, such as DENDRAL (Feigenbaum 1971) or MYCIN (Shortliffe 1976), lies in an explicit distinction between the knowledge useful for solving a class of problems and the programs which will use it, instead of the classical procedural implementation of knowledge under an implicit form in an algorithm. This fundamental characteristics give attractive properties to expert systems, particularly :

- possibility of dealing with uncomplete or even conflictual data,
- ease of examining and of updating the knowledge base. That makes it possible to incrementally develop large systems,
- possibility of explaining the reasoning line. This point is important both for debugging a system and then for teaching new humans experts.

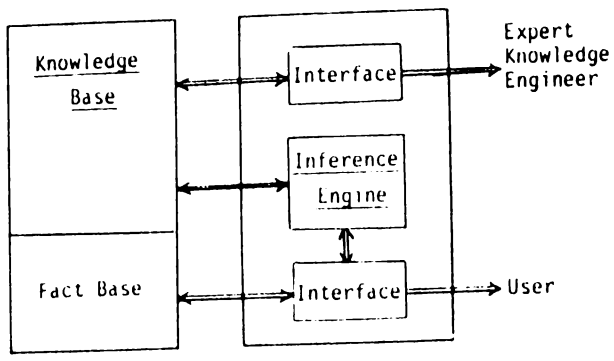
2.2. Architecture of an Expert System

An expert system is basically made up of three separate parts, i.e.

- a sub-system for entering and modifying the knowledge base that ensures the communication between the system and the human expert who gives his expertise,
- the expert system kernel which comprises :
  - . a knowledge base made up of facts about the problem and of the knowledge and metaknowledge itself,
  - . an inference engine, i.e. a procedure which manipulates the knowledge base,
- an interface with the system user which serves both for consulting the system and for teaching purposes. This third part may be very simple in case

of expert systems directly connected with an industrial process.

Figure 1 summarizes the architecture of a typical expert system.



Architecture of an Expert System
Figure 1

This architecture may vary to a certain extent according to the knowledge representation scheme that has been adopted. A commonly used scheme consists of production rules (cf. section 4.) which represent small, independent chunks of knowledge. This paradigm can be found in numerous expert systems, but it is sometimes insufficient for representing complex, structured knowledge. Frames and object oriented representations are used in this case as we will see later on.

2.3. Applications

Expert systems have proven to be applicable in a variety of domains. Some basic guidelines can be used for choosing an application : the problem should be technical enough and not include too much common sense knowledge ; it should be of medium difficulty, clearly defined and well bounded ; a human expert must be able to solve this problem and has to be available for the duration of the project since a fundamental point is the transfer of expertise from the human expert into the system.

Here are some typical kinds of activity most present and forthcoming expert systems correspond to :

- data interpretation : in various fields such as geology (PROSPECTOR (Duda 1978)), mass spectroscopy (DENDRAL), laser spectroscopy (EXSYLA (Haton 1985)). In more complex cases (e.g. for speech and image data) it becomes necessary to develop sophisticated architectures that will be described in the following sections of this chapter,

- diagnosis : this domain has been extensively studied and corresponds to a large number of practical systems as well in medicine (MYCIN and several other systems) as for machine or computer failure diagnosis (CATS-1 at General Electric, DART at IBM, etc.),

- control and monitoring : this is a particular case of diagnosis with the difficulty of having to reason about time evolution of data,

- prediction : expert systems can complement classical methods like simulation. The problem consists of predicting the future behaviour of a system in such fields as economy, politics, meteorology, finance, etc.,

- planning : this task is also usually carried out by classical computer techniques like operational research but expert systems may be in some cases more flexible or efficient, though not optimal,

- design : the problem consists of creating or assisting the creation of given objects, e.g. VLSI chips or house maps.

We will briefly review some typical expert systems in order to see the basic principles involved in their design.

MYCIN is a medical system that diagnoses of bacterious infections and recommands therapy. The knowledge base is made up of production rules which are used by the inference engine in a strictly backward chaining mode (see section 4.). Each rule is given a confidence factor which makes it possible for MYCIN to carry out an approximate reasoning based on a generalized Bayesian model. The following is a typical production rule in this system (translated from the internal LISP formalism into pseudo-natural language :

IF The infection which requires therapy is meningitis,
AND The patient has evidence of a serious skin or soft tissue infection,
AND Organisms were not seen on the strain of the culture,
AND The type of infection is bacterial,
THEN There is evidence that the organism (other than those seen on cultures or smears) which might be causing the infection is staphylococcus-coagpos (.75) or streptococcus (.5).

In MYCIN all the knowledge is coded into rules or meta rules. In several more recent systems rules are complemented by other kinds of representation, e.g. a semantic network embedding geological knowledge in PROSPECTOR or tree-like structures of parameters in TOM, an expert system for diagnosing tomato plant disease developed by Cognitech in Paris.

XCON (initially developed at Carnegie-Mellon University under the name of R1 (Mc Dermott 1980)) is able to configuring VAX computer systems from a customer's order. This system is now in daily use at DEC and comprises more than three thousand rules. That corresponds to the largest knowledge base used in a commercial expert system. XCON basically operates in a forward chaining manner by trying to match the IF conditions of a rule to the current situation. A typical production rule is for instance :

IF The most current active context is assigning a power suply,
AND An SBI module of any type has been out in a cabinet,
AND The position it occupies in the cabinet (its nexus) is known,
AND There is space available in the cabinet for a power supply for that nexus,
THEN Put the power supply in the cabinet in the available space.

2.4. Present Limitations and Further Development of Expert Systems

Current expert systems present strong limitations with respect to several aspects :

- the behaviour and the performances of a system : when compared to humans experts, expert systems are still very narrow and limited and, moreover, they are usually not able to determine when a problem is beyond their capacity or even completely outside their field of expertise. Very few systems have learning capacities although several research projects are in progress in this field,

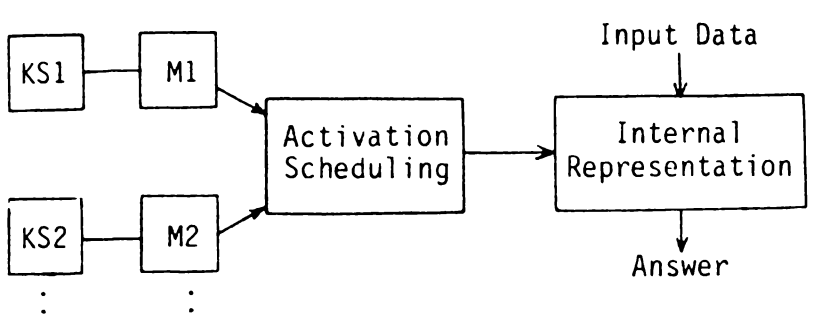
- the construction of a system : the extraction of knowledge from a human expert is presently laborious and must be conducted by a knowledge engineer (despite of the existence of sophisticated software tools this operation cannot be done directly by the expert).

Even with these limitations, expert systems have proven to be very useful in very diverse domains of activity. Many expert systems are under development and the number of systems actually used outside the laboratories is increasing rapidly. Future systems will feature deeper reasoning capabilities and various

knowledge representation schemes appropriate to a specific problem. We will for instance see in the following sections that sophisticated models like the blackboard model make it possible to combine rule-based and other kinds of reasoning.

3. BASIC PRINCIPLES OF UNDERSTANDING SYSTEMS

The first step in the design of an understanding system consists of collecting the various relevant knowledge sources (KS) and of designing a structure for the efficient cooperation of these KSs. Figure 2 illustrates the situation, with several knowledge sources, KSi, each having its own activation mechanisms, Mi.



Overall blockdiagram of an understanding system

Figure 2

This structure is applicable for several domains of AI where the complexity of the task makes it mandatory to use several types of information, such as speech understanding vision, natural language processing or signal interpretation.

For instance, in vision, the localization and recognition of objects on an image - or a fortiori on a 3D scene - is not only a problem of matching some parts of the scene against stored prototypes, since problems of object-orientation, illumination, etc. can cause important variations in the perception of a given object. A classical solution consists of representing an object or an image as a relational graph. These graphs are then compared during the processing of a scene. Similar methods are used for the representation of words in a lexicon.

A very common strategy used in these fields relies on the so-called hypothesis-and-test or prediction-verification paradigm. In this strategy, hypotheses are emitted at the various processing levels according to both the available knowledge and to various indices. These hypotheses are then verified or concelled by other knowledge sources. We will often encounter this strategy in this chapter.

A large amount of our knowledge about speech comes from linguistics. The speech signal is encoded at various levels during the speech production process and the automatic decoding will have to take into account the corresponding KS :

- = acoustics : for handling signal preprocessing and feature extraction,
- = phonetics, which is related to the transcription of the speech signal into discrete phonetic units,
- = phonology : together with phonetics, this component deals with alterations of sounds (accent, etc.) and with contextual variations (liaisons, assimilations, co-articulations, etc.),
- = prosody : these features are specific to speech communication and highly important in speech understanding for a human listener,
- = lexicon : the word is an essential element of sentence structure,
- = syntax : related to the structure of a message according to the grammar of the language,
- = semantics : representing the meaning of words and conceptual dependencies,
- = pragmatics : more specific than semantics, this KS is associated with the context of the application.

Most of these KSs also play a role in written language understanding. Results obtained in this domain can therefore be re-used to a certain extent in speech understanding. However the specificity of oral language and the indeterminism which appears during the understanding process makes it necessary to design new models or at least to adapt them.

Indeterminism is inherent to most domains of signal understanding. It basically comes from two sources :

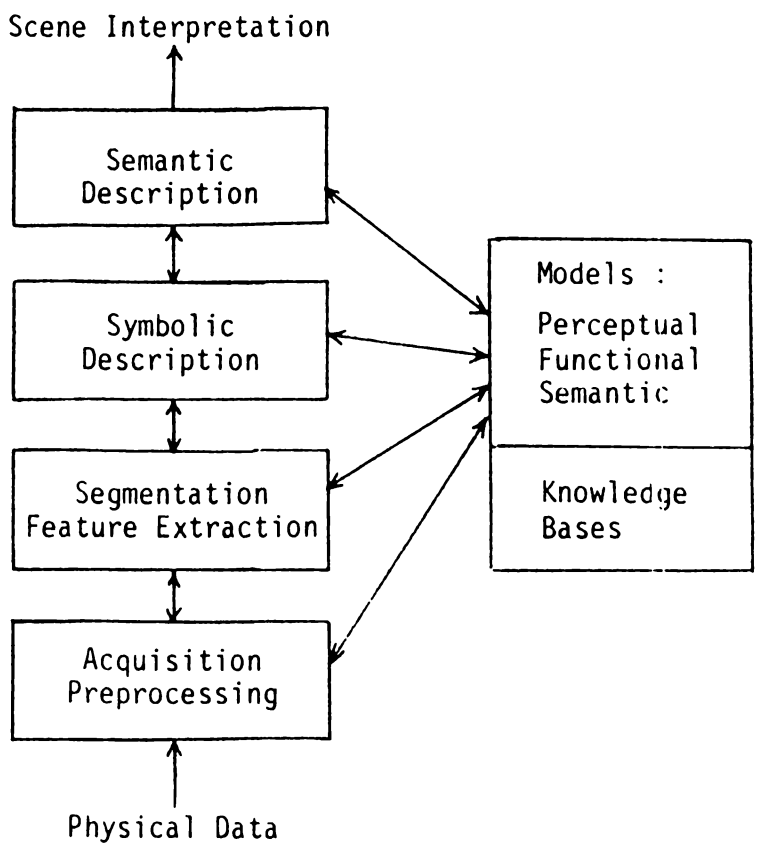
(i) errors in the identification of low-level primitives (phonemes in speech or basic patterns in an image), due to the variability of signals and to noise,

(ii) errors in high level interpretation, due to the continuous character of signals (erroneous spotting of a word in a sentence or of an object in a scene), or to ambiguities and imprecisions in the KSs (for instance prosody obviously plays an important role in speech understanding but no theory has yet been able to formalize this KSs satisfactorily).

These different errors are then propagated throughout the interpretation process and thus make the problem a very difficult one.

As far as speech is concerned it is important to notice the fundamental role played by words and the lexical level. Word hypotheses are emitted in both a bottom-up or data-driven mode from the acoustic data and in a top-down or model driven mode from context and high level KSs.

A similar situation exists in computer vision, except that the linguistic models available for speech are more operational than those for vision. A computer vision system can be described (figure 3) as a set



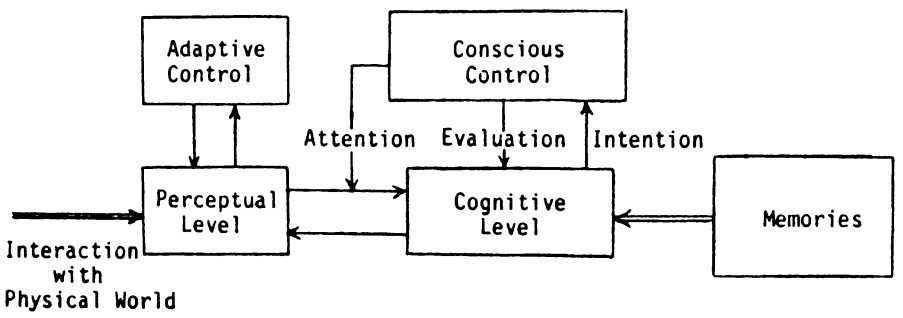
Principle of a computer vision system

Figure 3

of interrelated processes which progressively transform sensory data in accordance with different models (perceptual, fonctional, etc.) that incorporate the knowledge available at the various levels in order to yield a symbolic and finally semantic description of the scene.

In this case the interaction between low level and high level processing involves objects which play a similar role to that of words in speech. We will see later on that very similar models have been used in speech understanding and in vision.

It is important to note that the distinction between perceptual or low level (i.e. pattern recognition) and cognitive or high level (i.e. artificial intelligence) is only used for the sake of simplicity. In fact, during the human understanding process there is a very high degree of interaction between the sensory organs and the brain. Figure 4 is an attempt to summarize this situation. This perceptual, or rather sensori-motor, level is controlled by an adaptive system which mostly works in an unconscious way. It is tightly coupled with the cognitive level and the memories. This latter level is controlled by a conscious system which corresponds to some of the fundamental mechanisms of audition and vision such as evocation, intention or attention. These mechanisms are not yet very well understood, and attempts to implement them in automatic systems have been far from satisfactory.

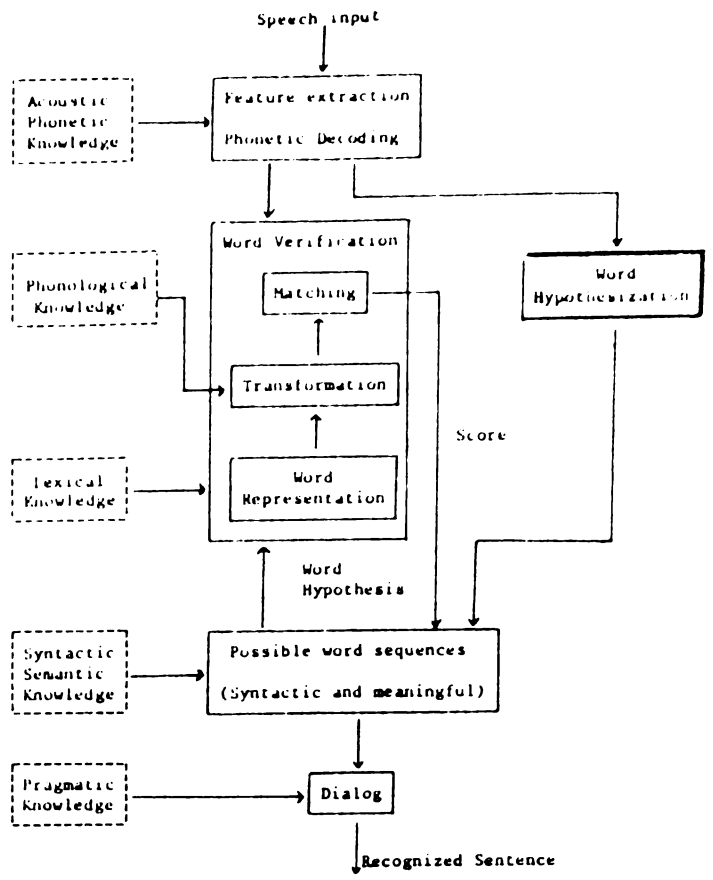


Schematic block-diagram of human understanding process
Figure 4

One way of characterizing low level and high level processing is in terms of knowledge available. High level processing carries out reasoning by taking into account the various domain-specific knowledge sources (syntax, semantics, etc. in case of speech, or symbolic and semantic description of objects in case of vision). Low level processing, on the other hand, works directly on physical data with general models which are usually less dependent on the domain of application (signal processing techniques, segmentation, etc.).

In the case of speech, the central role of the lexical level is clearly illustrated in figure 5. This figure in no sense represents the architecture of a speech understanding system ; rather it indicates the various processes involved in the operation of such a system and the roles of the various KSs concerned.

Most of complex understanding systems can be similarly organized into successive levels corresponding to a progressively more and more symbolic description of initial data. In order to avoid a combinatorial explosion of possible solutions, several techniques are used at each level for decreasing the ambiguities. Such techniques do not always come from the AI field. For instance, dynamic programming is widely used in speech understanding for spotting speech units precisely (syllables, words) within a sentence. Similarly, relaxation techniques make it possible to improve the results of low-level image processing : contour extraction, etc. (Zucker 1976). These techniques can also be used for solving ambiguities at a higher level of abstraction, e.g. in object labeling (Barrow 1976).



Typical processes in continuous speech recognition
Figure 5

4. KNOWLEDGE REPRESENTATION AND USE

4.1. General Models

The performances of an understanding system are highly dependent on the choice of efficient representation schemes for the various KSs. There exist two extreme solutions :

- the first is to define a unique structure in which all available knowledge will be integrated,
- the second is to keep the various KSs totally independent, and hence increase the modularity of the system.

Representing knowledge in a single structure is the solution that was chosen in the HARPY speech understanding system (Lowerre 1976). In this system, all available KSs - from acoustics to semantics are precompiled into a huge network which contains all the phonological variations of the various sentences allowed by the grammar of the language. The recognition of a sentence consists of finding the optimal path in this network. HARPY uses a heuristic beam search strategy which retains only a small number of the best solutions, at each step thus avoiding a combinatorial explosion. This strategy is a sub-optimal solution of a dynamic programming algorithm initially developed in the DRAGON system. A similar solution has been implemented in the ARGOS system for computer vision (Rubin 1978). HARPY was the system which best matched the initial requirements of the ARPA Project on Speech Understanding (Newell 1973) : precompiling knowledge is definitely a more efficient solution than using it in an interpretive mode. However this solution is very ad hoc and presents important drawbacks. First, every change in the language (or in the user) makes it necessary to recompile the network. That always means a lot of work, even though it can be partly automatized. Secondly, this solution is only tractable for simple situations, e.g. in highly constrained languages or simple and repetitive scenes. For instance it would be impossible to represent in this way all the KSs necessary for understanding pseudo-natural languages.

The second extreme solution consists in defining

a system architecture in which each KS behaves completely independently of all the others but can communicate with them. However, a completely heterarchical model with connection between all pairs of KSs would be too complex to implement. The blackboard model represents a good compromise. In this model, the different KSs are independent processes which communicate with each other only through a common database called blackboard. The HEARSAY II speech understanding system (Lesser 1975), for instance, has been built on this model.

The role of the blackboard is twofold :

- it ensures the passage of a message (i.e. an hypothesis) from one KS to another : every KS asynchronously emits hypotheses at its own level of expertise (phoneme, syllable, word, phrase, etc. for speech) and posts them on the blackboard. A hypothesis emitted by a particular KS Si may activate another KS Sj. The activation of a KS is thus pattern-directed : certain preconditions have to be fulfilled in the blackboard in order to activate a given KS. This notion has been very fruitful in AI (Waterman 1978). We will come back later on to this "Condition-Action" structure of KSs in HEARSAY II. In such cases, the action of a KS consists of creating, modifying or cancelling one or several hypotheses on the blackboard,

- it contains the partial interpretations of a sentence in terms of sets of hypotheses.

The blackboard model is general, and has been applied to various domains of AI : computer vision (Prager 1977) (Nagao 1979), signal interpretation (Nii 1978), cristallography (Engelmore 1979). This is not surprising if one considers the analogies which can be found between domains like speech or vision which were mentioned above. It is interesting to notice that there exist large similarities in the structure of the blackboard of the HEARSAY II system and of the VISIONS system for image understanding (Hanson 1978). However, there is an important difference in that the blackboard data base in HEARSAY II has a time dimension (speech is a time varying process) whereas VISIONS is only able to process static scenes.

Several intermediate solutions have been used for representing knowledge in understanding systems. The most usual model is a hierarchical one operating under the control of a supervisor, instead of the asynchronous, pattern-directed control of the blackboard model. Systems like HWIM (Woods 1976), ESOPE, KEAL, LITHAN, MYRTILLE I and II in speech understanding are good illustrations of this model. We will see in the next section the control structures which can be associated with this solution. Similar exemples could be given from other fields of understanding.

The idea of integrating a certain amount of knowledge by precompilation is also often used. For instance, phonological knowledge which is responsible for phonetic alterations of words in context can be represented in terms of contextual rewriting rules which apply to word entities in the lexicon (Shoup 1980). This application of phonological rules to basic word-forms in the lexicon can be done either during the understanding process at each access to the lexicon or all at once in a single, preliminary phase. This phase consists of a precompilation of the set of phonological forms of the words. This latter solution is much more efficient as far as the computation time is concerned and is used in several large systems for speech understanding, such as HWIM.

Similarly, the use of some kinds of precompiled knowledge increases the efficiency of vision systems. The MIRABELLE system (Masini 1978) we have developed in our group uses a compiler of descriptions for interpreting handwritten drawings. In a preliminary phase this compiler extracts information from the structural description of the class of drawings to be recognized. This information is then used during the recognition process in order to control the analysis. We have extended this idea to the interpretation of 3D scenes in the TRIDENT vision system (Masini 1984). Precompiling a knowledge base (e.g. a set of production rules) also contributes to efficiency, as for example in the PROSPECTOR system (Konolige 1979).

4.2. Knowledge-Based and Expert Systems

We have already noted the present tendency in AI to design systems which are able to solve problems in very narrow domains by using knowledge specific to these domains. In such knowledge-based systems a very common paradigm for representing knowledge is the production rule paradigm (Davis 1977).

Let us briefly recall the basic principles of productionrule systems. A production rule is of the form : IF Condition THEN Conclusion
where "Condition " represents a conjunction of predicates which have to be verified to order to enable the rule to be fired. The application of this rule results in "Conclusion", i.e. emission or modification of an hypothesis or a fact. This way of using a rule is referred to as forward- or data-driven chaining. It is also possible to use a rule in a backward-chaining mode. It consists of taking the "Conclusion" part of a rule as a goal to be reached and of considering the verification of the predicates in the left-hand side of this rule as new sub-problems.

A production system can be regarded as a simple example of a pattern-driven system like HEARSAY II, even though the knowledge sources in HEARSAY are much more complex. This system has been rewritten as a pure production-rule system (Mc Craken 1981).

One model proposed for knowledge utilization in human brain has been based on the idea of very rapid and frequent transfers between a long-term memory which stores knowledge and a short-term memory which uses it. Production rules can be considered as implementing these knowledge transfers (Newell 1976).

An important characteristic of knowledge-based and expert systems is that they represent a large amount of knowledge by breaking it down into small chunks, such as production rules, which are easy to formalize and to manipulate. This interesting property can be taken into account in understanding systems for solving specific problems, provided that human experts are able to solve these problems and to formalize their expertise. This is particularly important for areas in which the human expertise involves some kind of visual reasoning about data (vision, signal interpretation, etc.). As an example we will now present the case of acoustic-phonetic decoding of speech or, more precisely, of speech-spectrogram reading.

Present automatic phonetic decoders only achieve about 65 to 70% of correct recognition for a single speaker. They are all based on a more or less sophisticated phonetic model, for instance :

- a set of reference patterns which are tentatively matched against speech segments (in a pattern-recognition approach to the problem),

- a set of hierarchical decision rules (in the AI approach),

- a homogeneous stochastic model, e.g. a Markov model (in the information theory approach) which makes it necessary to process very large amount of speech data in order to tune the parameters of the model.

It is worth noticing that these models are in no way incompatible : a satisfactory solution might consist of combining rule-based reasoning with a certain amount of (eventually stochastic) pattern matching.

Rule-based models have proven reasonably efficient, particularly when one can incorporate in the rules the human expertise accumulated through the examination of a large number of cases. One way of capturing such an expert knowledge consists of considering the activity of a phonetician while reading a speech spectrogram (Zue 1979). The acquisition of this expertise, and more generally of differents kinds of knowledge about speech, will certainly help significantly in improving the segmentation and phonetic labeling of speech (Memmi 1983) (Caelen 1983) (Gillet 1984) (Meloni 1985) (Migozuchi 1984).

The problem of spectrogram reading is conceptually difficult, since it combines cognitive reasoning processes with the perceptual aspects of visual inspection. However the expression of this problem in terms of image processing and vision is quite unhelpful.
Moreover, recent studies (Bush 1983) have shown that an expert is able to adapt his expertise in order to decode not only "classical" spectrograms but also LPC spectra (Makhoul 1975) or tables of numerical values. The central question is, therefore, to capture the knowledge and metaknowledge that enable the expert to develop sophisticated decoding strategies. In the SYSTEXP project we are developing in Nancy at present (Carbonell 1984), we have put particular emphasis on this transfer of knowledge, which turns out to be more difficult than in more usual expert systems. Expert system technology provides a powerful tool for improving our knowledge of the speech decoding process, even though, for reasons related to the efficiency of the machines available at present, the final implementation of a speech recognition system will probably take the form of a procedural system (Haton 1984).

The expertise of the phoneticiens can be formalized into classical production rules of various kinds :

- phonetic class identification rules,
- exclusion rules,
- contextual rules (the most common ones), for instance :
  . IF Right context = /i/
  . AND Formant 2 is increasing
  . AND Formant 3 is increasing
  . AND First formant visible above 1000 hz = 2200 Hz
  . AND No discontinuity with formant 2 of /i/
  . THEN /m/
- meta-rules expressing the choice of various strategies according to the actual situation.

These rules are also useful in speech recognition tasks other than those involving phonetic decoding. We have, for instance, implemented a 2000 isolated word recognizer which uses the same expert knowledge base for the selection of small subsets of phonetically similar words (Mari 1984).

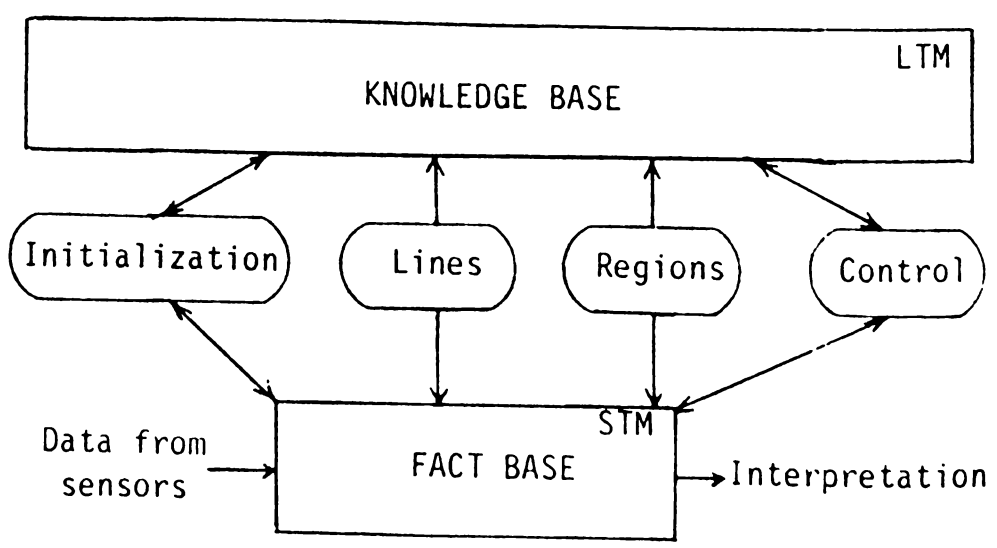
A knowledge-based approach has also been used in the field of pitch detection (Dove 1983). In this case knowledge engineering once again provides an efficient framework for mixing signal processing methods and human expertise.

The segmentation of the speech wave into phonetic units can also largely benefit from a knowledge-based azpproach. This segmentation can be carried out either by merging together small elements of speech (e.g. centiseconds) or by finding boundaries in the speech continuum. A similar alternative exists in image segmentation, where we can merge pixels or small parts according to their similarity, or progressively split an image into smaller regions. Image segmentation can thus be solved by an expert system approach (Nazif 1984). A common characteristic if the two domains is the non-unicity of the segmentation obtained. That makes further processing more complicated and necessitates the handling of multiple partial solutions throughout the understanding process.

The formalization and exploitation of phonological constraints can also be thought of in terms of knowledge-based systems (Oshika 1984). A fundamental phase consists of collecting an exhaustive base of rules representing the morpho-phonological phenomena of the language (Pérennou 1982).

More generally the problem of understanding complex "objects" as encountered in the domains of speech or vision can be solved by knowledge-based systems. For instance, expert systems have been designed for the image interpretation of aerial photographs (Mc Keown 1983). When the complexity of the tasks increases it becomes necessary to use a number of different knowledge bases and, therefore, a single expert system is no longer sufficient. Several models have been proposed, both for object identification (Kim 1984) (Reynolds 1984) and for the more general problem of scene interpretation (Riseman 1984) (Barrow 1976). We have already seen the interest of the blackboard model at this level. That model represents an efficient framework for an asynchronous, partially data-driven solution to the understanding problem. It has a number of important characteristics, including real cooperation between processes, the possibility of combining partial solutions for incrementally building up the interpretation of complex objects, problem solving at various levels of abstraction, etc. These characteristics are important in all domains of knowledge engineering and therefore a number of tools for building knowledge-based systems have been founded on them, such as HEARSAY III (Balzer 1980) or AGE (Nii 1979).

In fact the blackboard model is not the only one that has been proposed, particularly for KS cooperation. For instance, in the expert society model each expert has its own internal reasoning logic and control structures and communicates directly with the other expert modules (cf., for instance, (De Mori 1983) for an application to speech understanding). The model proposed by Nazif and Levine in their vision system (Nazif 1984) is a slightly different version of the blackboard model. In their system (figure 6) various processes responsible for different understanding tasks such as system initialization, line processing, region processing, general control, etc. communicate through two different blackboards :



A vision system based on a two-blackboard model
Figure 6

- a short term memory (STM) which contains the data and facts about the actual problem (initial data, segmentation results and final interpretation),
- a long term memory (LTM) which contains knowledge and metaknowledge in the form of rules.

5. SEARCH STRATEGIES AND CONTROL STRUCTURES

In order to use various KSs efficiently, it is necessary to implement sophisticated control structures for controlling the interpretation process. Moreover, the combinatorial explosion of solutions has to be reduced by means of appropriate search strategies. We will now discuss these points and show their influence on the overall architectures of understanding systems.

5.1. Models for Control Structures

As we have seen, indeterminism and uncertainty are inherent to understanding problems. One way for coping with this consists of using hypothesis valuation techniques and of propagating "confidence factors" during the interpretation process. The implementation of sophisticated control structures is also of primary importance. There are two basically different philosophies for control structures :

- bottom-up or data driven control, which emits hypothesis in accordance with the input data and tries progressively to build up the interpretation of the object (sentence, scene, situation, etc.) under consideration,
- a top-down or model driven control, which consists of emitting hypotheses about the input data by taking into account high level knowledge available. This type of control can be associated with the classical AI notion of problem reduction.

When the KSs are coded under a Condition-Action form (like production rules or the blackboard model) these two control structure classes correspond to a forward and a backward chaining respectively.

In fact, the two methods refer to the same KSs and manipulate the same intermediate notions (e.g. phrases, words, syllables, etc. for speech or objects, regions, lines, etc. for images) and they mainly differ in the order in which the different processings are carried out and in the degree of detail into which they enter. The notions of bottom-up and top-down control are an oversimplification of the implementation of processing levels interaction in actual systems. In the human brain the control structures reach still much higher level of sophistication (Geschwind 1980).

In a bottom-up speech recognition system, a word lattice containing all the possible occurrences of words within a sentence is built up from the acoustic data and the lexical knowledge. Syntactic and semantic KSs are then used to select the most plausible sequence of words that will represent the actual interpretation of the sentence. This method is interesting, since it is not too sensitive to noisy data. However it becomes untractable when the size of the application (number of words, complexity of the language) increases, due to the large number of possible solutions. The same situation exists in vision : a purely bottom-up approach consisting of starting from pixels in order to build up to a scene is only valuable in simple industrial applications where adequate illumination conditions make it possible to only manipulate binary images.

In top-down approaches, high level knowledge about the application universe is used for making assumptions about the constituents of a sentence or of a scene. The predictive aspect of top-down control makes it possible to cancel a large number of hypotheses. This point is particularly interesting in large applications where there is a high number of possible combinatorial solutions. On the other hand, a purely top-down control is relatively more affected by noise in the incoming data.

The representation of knowledge in frames provides an efficient way of implementing top-down control. Frames were initially proposed for natural language processing (Schank 1975). They represent for instance models of classes (Brachman 1978) and correspond to a less fine granularity of knowledge than production rules. This representation scheme features the important notion of procedural attachment, a notion that makes it possible to activate procedures when the interpretation process needs it. We have used this approach for controlling the acoustic-phonetic decoding of speech in a knowledge-based system (Haton 1985). A somewhat different, but related, solution was proposed by (Green 1984). It consists of defining an intermediate representation structure ("Speech Sketch") in terms of frames and of trying to instantiate a frame prototype within a sketch. The interpretation of other kinds of objects, e.g. ECG signals (Lee 1984) can also be approached in this way as well as the classification of objects in a vision system. The use of representation structures in vision, as initially suggested by (Marr 1982), seems as promising in this field as it does for speech.

Frames represent one particular case of object-oriented representation of knowledge. The use of so-called object-oriented languages is of interest to understanding systems from at least two points of view :

- they provide the software environment for representing in a structured manner and for using the different entities which are manipulated (phonetic and supraphonetic units, objects of a scene, etc.),
- the basic mechanism of message-passing that underlies all these languages is well suited to the cooperation of KSs on a prediction-verification scheme cf., for instance, the cooperation of KSs in the blackboard model by emission of hypotheses, i.e. messages. It is moreover adapted to a parallel hardware implementation.

For simple applications an understanding system may operate in a structly bottom up way. On the other hand, it is very rare to find purely top down systems. An exemple is one version of the HWIM speech understanding system based on Klatt's analysis-by-synthesis model (Klatt 1975). In this version, the top down assumptions about words are pushed down to the level of acoustic signal by means of a speech synthesis system. A word assumption is then verified by dynamic comparison of its acoustic spectrum with a position of the input sentence. This solution is not computationnally efficient but it is interesting at the level of principle. One version of the ESOPE speech understanding system (Mariani 1982) is also based on a similar idea.

In most cases, systems based on top-down control structures incorporate some bottom-up process. The most commonly encountered arrangement consists of deriving a bottom-up symbolic description of the input data that corresponds at a same time to an important reduction of these data. This description might take the form, for exmaple, of a phoneme lattice in the case of speech or a set of lines, regions or partially identified objects in vision. These preprocessed data are then taken into account by top-down processors.

As far as speech understanding goes, bottom-up or partially top-down systems are almost equally efficient for processing sentences of artificial, highly constrained languages (see, for instance, the MYRTILLE I system (Pierrel 1975) for a typical example). But for pseudonatural languages it is absolutely necessary to implement much more

sophisticated control strategies that combine bottom-up and top-down methods. Such structures can be found in the systems we have already mentioned : HEARSAY II, HWIM, MYRTILLE II, etc.

It is interesting to consider the development of understanding systems during the past fifteen years, both in vision and in speech. The first systems that were built relied heavily on a bottom-up methodology which consisted of extracting the maximum amount of information from the data by the use of powerful but blind techniques of signal processing and pattern recognition.

As we have already seen, except in simple cases (binary images or artificial sentences), such methods cannot handle the combinatorial explosion of possible solutions. They were therefore abandoned and replaced by top-down methods and then by mixed methods for understanding as in the case of the ACRONYM system (Brooks 1981). This system is based on a geometrical description of objects which is independent of the application domain. However, these models are not yet completely satisfactory and there is now a tendency towards basic studies about the physical and physiological phenomena that are involved for instance in the process of image or speech production and perception : properties of objects (luminosity, brightness, textures, geometry of surfaces, etc.), vision process (Brady 1981), etc. on the one hand, phonetic and phonological properties of sounds, phonation and audition processes, etc. on the other. This promising new tendency calls on a number of very different disciplines and necessitates the elucidation of diverse knowledge and expertise domains.

5.2. Search Strategies

At each step during the understanding process a potentially large set of partial solutions has to be examined. In order to restrict this set to a computationnally reasonable size and to avoid any combinatorial explosion, sophisticated search strategies have to be implemented so that an acceptable interpretation of the incoming data -though not always the optimal one is finally reached. The search problem is well known in AI and it has been addressed a great number of times. Search strategies are crucial in complex problems like speech understanding or vision since the solution space is very large, whereas in more restricted tasks such as diagnostic expert systems (e.g. the MYCIN system (Shortliffe 1976) which was among the first systems developed) an exhaustive search throughout the solution space is practicable.

Dynamic programming yields an optimal solution to the search problem. For instance, it has been implemented by IBM (Jelinek 1982) in association with a Markov model of the language by using the Viterbi algorithm. Relaxation techniques used in image processing are approximately equivalent to this algorithm.

Dynamic programming algorithms have a complexity proportional to n^2 where **n** represents the number of states to be explored. They are therefore often much too time-consuming, in which case it is necessary to use suboptimal but faster heuristic search strategies.
This means that, instead of a global strategy, we have a sequence of local decisions which will hopefully lead to a global acceptable solution in accordance with the general prediction-verification paradigm. Two widely used strategies are :

- a best-first search strategy with backtracking : it consists of keeping the most promising solution at each step and of developing it as far as possible. If this becomes impossible, a backtracking algorithm is activated for considering another solution. This strategy can be found in numerous systems, since it is easy to implement and efficient for small-and medium-size applications,
- a beam-search strategy : this consists of simultaneously retaining the **k** best solutions and developing them in parallel, without any backtracking. This strategy is usually more efficient than the best first strategy, especially in the case of speech understanding (Quinton 1982). It is the strategy used in the HARPY system.

These two strategies are very popular in various domains of AI and have been adapted in a number of different ways. Specific constraints of speech and vision have led to the design of a new class of search strategies, called anchor-point or island-driven strategies. The basic idea consists of starting the interpretation of a sentence or an image from one or several reliable anchor-points. Such strategies are rather complex to implement but prove to be very useful for processing pseudo-natural sentences or images and handwritten sketches (Haton 1976).

6. CONCLUSION

Knowledge-based techniques have led to significant successfull applications of AI in various domains. We have shown in this chapter the importance of such techniques in computer understanding systems, especially in the fields of speech and vision. These fields are characterized by a close interaction between the perceptual and cognitive features and by a high degree of indeterminism. The interpretation of "objects" in this framework makes it necessary to arrange for several knowledge-sources to cooperate in an efficient manner. We have particularly emphasized the representation of knowledge and the way it can be used together with the associated control structures and search strategies. Practical examples have shown the analogy that exists between the two domains of speech and vision.

Considerable progress has been made in computer understanding during the past ten years even though the models that have been proposed still need improving. An important point is the necessity of adapted, parallel architectures for real-time implementation of these models. The different projects on new computer generation and the use of VLSI custom circuits will hopefully bring solutions to this problem.

REFERENCES

(Balzer 1980) R. Balzer et al., HEARSAY III : A Domain Independent Framework for Expert Systems, Proc. Conf. Amer. Assoc. Intell., 1980.

(Barrow 1976) H. Barrow and J. Tenenbaum, MSYS : A System for Reasoning about Scenes, SRI Techn. Note 121, 1976.

(Brachman 1978) R.J. Brachman, A Structural Paradigm for Representing Knowledge, BBN report 3605, 1978.

(Brady 1981) J.M. Brady, The Changing Shape of Computer Vision, Artificial Intelligence, 17, n° 1-3, pp. 1-15, 1981.

(Brooks 1981) R.A. Brooks, Symbolic Reasoning among 3D Objects and 2D Models, Artificial Intelligence, 16, pp. 285-348, 1981.

(Bush 1983) M.A. Bush, G.E. Kopec and V.W. Zue, Selecting Acoustic Features for Stop Consonant Identification, Proc. IEEE ICASSP, Boston, 1983.

(Caelen 1983) J. Caelen et al., Structuration des informations acoustiques dans le projet ARIAL, Speech Communication, 2, n° 2-3, pp. 219-222, 1983.

(Carbonell 1984) N. Carbonell, D. Fohr, J.P. Haton, F. Lonchamp, J.M. Pierrel, An Expert System for the Automatic Reading of French Spectrograms, Proc. IEEE ICASSP-84, San Diego, 1984.

(Davis 1977) R. Davis and J. King, An Overview of Production Systems, Machine Intelligence, 8, pp. 300-332, 1977.

(De Mori 1983) R. De Mori, P. Laface, G. Petrone and M. Segnan, Access to a Large Lexicon Using Phonetic Features, EUSIPCO, Erlangen, 1983.

(Dove 1983) W. Dove et al., Knowledge-Based Pitch Detection, Proc. IEE ICASSP, Boston, 1983.

(Duda 1978) R.O. Duda et al., Development of the PROSPECTOR Consultation System for Mineral Exploration, SRI Int. Report, 1978.

(Engelmore 1979) R.E. Engelmore and A. Terry, Structure and Function of the CRYSALIS System, Proc. IJCAI-6, 1979.

(Feigenbaum 1971) E.A. Feigenbaum et al., On Generality and Problem Solving : A Case Study using the DENDRAL Program, in B. Meltzer and D. Michie editors, Machine Intelligence 6, Edinburgh, University Press, 1971.

(Geschwind 1980) N. Geschwind, Neurological Knowledge and Complex Behaviors, Cognitive Science, 4, n° 2, pp. 185-193, 1980.

(Gillet 1984) D. Gillet et al., SERAC : Un système expert en reconnaissance acoustico-phonétique, 4ème congrès RF-IA, AFCET-INRIA, Paris, 1984.

(Green 1984) P.D. Green and A.R. Wood, Knowledge-Based Speech Understanding : Towards a Representational Approach, Proc. ECAI-84, pp. 337-340, 1984.

(Hanson 1978) A.R. Hanson and E.M. Riseman, VISIONS : A Computer System for Interpreting Scenes, in Computer Vision Systems, A.R. Hanson and E.M. Riseman, editors, Academic Press, 1978.

(Haton 1976) J.P. Haton et R. Mohr, A New Parsing Algorithm for Imperfect Patterns and its Applications, Proc. ICPR, San Diego, 1976.

(Haton 1984) J.P. Haton and M. Lazrek, Segmentation et identification des phonèmes dans un système de reconnaissance automatique de la parole continue, 4ème congrès RF-IA, AFCET -INRIA, Paris, janvier 1984.

(Haton 1985) J.P. Haton and J.P. Damestoy, A Frame Language for the Control of Phonetic Decoding in Continuous Speech recognition, Proc. ICASSP, Miami, Florida, 1985.

(Haton 1985) J.P. Haton et al., EXSYLA, un système expert pour l'interprétation de spectres de masse en spectrométrie laser, Proc. 5th Int. Workshop on Expert Systems and their Applications, Avignon, 13-15 May, 1985.

(Jelinek 1982) F. Jelinek, Self-Organized Continuous Speech Recognition, in Automatic Speech Analysis and Recognition, J.P. Haton, editor, D. Reidel.

(Kim 1984) J.H. Kim, D.W. Payton and K.E. Olin, An Expert System for Object Recognition in Natural Scenes, Proc. First Conf. on Applic. of AI, Denver, 1984.

(Klatt 1975) D.H. Klatt, Review of the ARPA Speech Understanding System, in Speech Recognition, D. Reddy, editor, Academic Press, 1975.

(Konolige 1979) K. Konolige, An Inference Net Compiler for the PROSPECTOR Rule-Based Consultation System, Proc. IJCAI-6, 1979.

(Lee 1984) H.S. Lee and N.V. Thakor, Frame-Based Understanding of ECG Signals, Proc. First Conf. on Applic. of AI, Denver, 1984.

(Lesser 1975) V.R. Lesser et al., Organization of the HEARSAY II Speech Understanding System, IEEE Trans. ASSP, 23, pp. 11-23, 1975.

(Lowerre 1976) B.T. Lowerre, The HARPY Speech Recognition System, Tech. Report, CMU, DEpt. of Computer Sciences, 1976.

(Makhoul 1975) J. Makhoul, Linear Prediction : a Tutorial Review, Proc. IEEE, 63, n° 4, pp. 561-580, 1975.

(Mari 1984) J.F. Mari, J.P. Haton, Some Experiments in Automatic Recognition of a Thousand Word Vocabulary, Proc. IEEE ICASSP, San Diego, 1984.

(Mariani 1982) J. Mariani, ESOPE ; un système de compréhension de la parole continue, Thèse d'Etat, Université de Paris VI, juillet 1982.

(Marr 1982) D. Marr, Vision, Freeman, 1982.

(Masini 1978) G. Masini, Réalisation d'un système de reconnaissance structurelle et d'interprétation de dessins, Thèse de 3ème cycle, Univ. de Nancy I, 1978.

(Masini 1984) G. Masini and F. Zaroli, Présentation de TRIDENT : un système d'interprétation d'images tridimensionnelles, 4ème congrès RF-IA, AFCET-INRIA, Paris, 1984.

(Mc Cracken 1981) D.L. Mc Cracken, A Production System Version of the HEARSAY II Speech Understanding System, Univ. Research Press, 1981.

(Mc Dermott 1980) J. Mc Dermott, R1 : An Expert in the Computer Systems Domain, Proc. Amer. Art. Intelligence Conf., 1980.

(Mc Keown 1983) D.M. Mc Keown Jr and J. Mc Dermott, Toward Expert Systems for Photo Intepretation, IEEE Conf. on Trends and Applications 83, pp. 33-39, 1983.

(Meloni 1985) H. Meloni, J. Gispert, J. Guizol, Traitement de connaissances pour l'identification analytique de mots dans le discours continu, Congrès AFCET Infoirmatique, Paris, 1985.

(Memmi 1983) D. Memmi, M. Eskenazi, J. Mariani, Un système expert pour la lecture de sonagrammes, Speech Communication, 2, n° 2-3, pp. 234-236, 1983.

(Mizoguchi 1984) R. Mizoguchi and O. Kakusho, Continuous Speech Recognition Based on Knowledge Engineering Techniques, Proc. 7th Int. Conf. Pattern Recognition, Montreal, 1984.

(Nagao 1979) M.T. Nagao et al., Structural Analysis of Complex Aerial Photographs, 6th IJCAI, 1979.

(Nazif 1984) A.M. Nazif and M.D. Levine, Low Level Image Segmentation : An Expert System, IEEE Tr. PAMI, 6, n° 5, pp. 555-577, 1984.

(Newell 1973) A. Newell et al., Speech Understanding Systems : Final Report of a Study Group, North-Holland, Amsterdam, 1973.

(Newell 1976) A. Newell, Production Systems : Models of Control Structures, in Visual Information Processing, W. Chase, editor, Academic Press, 1976.

(Newell 1983) A. Newell and H.A. Simon, GPS : a Program that simulates human Thought, in Feigenbaum and Feldman, editors, Computer and Thought, Mc Graw Hill, 1983.

(Nii 1978) H.P. Nii and E.A. Feigenbaum, Rule-Based Understanding of Signals, in Pattern-Directed Inference Systems, Waterman and Hayes-Roth, editors, Academic Press, 1978.

(Nii 1979) H.P. Nii and N. Aiello, AGE (Attempt to GEneralize) : A Knowledge-Based Program for Building Knowledge-Based Programs, 6th IJCAI, 1979.

(Oshika 1984) B. Oshika, Phonological Rules for Continuous Speech Recognition, in Towards Robustness in Speech Recognition, W. Lea, editor, Prentice-Hall, 1984.

(Pérennou 1982) G. Pérennou, Lexical Acess and Phonologic Processing in ARIAL II, 6th ICPR, Munich, 1982.

(Pierrel 1975) J.M. Pierrel, Contribution à la Compréhension automatique du discours continu, Thèse de 3ème cycle, Univ. de nancy I, 1975.

(Prager 1977) J. Prager et al., Segmentation Processes in the VISIONS System, Proc. 5th IJCAI, Cambridge, 1977.

(Quinton 1982) P. Quinton, Utilisation de contraintes syntaxiques pour la reconnaissance de la parole continue, TSI, 1, n° 3, pp. 233-248, 1982.

(Reynolds 1984) G. Reynolds et al., Hierarchical Knowledge Directed Object Extraction Using a Combined Region and Line Representation, Proc. Workshop on Computer Vision : Representation and Control, Annapolis, 1984.

(Riseman 1984) E.M. Riseman, A.R. Hanson, A Methodology for the Development of General Knowledge-Based Vision Systems, Proc. IEEE Workshop on Principles of Knowledge-Based Systems, Denver, 1984.

(Rubin 1978) S. Rubin, The ARGOS Image Understanding System, Ph. D. Thesis, Carnegie-Mellon University, Pittsburgh, 1978.

(Shank 1975) R.C. Shank, Conceptual Information Procesing, North-Holland, 1975.

(Shirai 1973) Y. Shirai, A Context Sensitive Line Finder for Recognition of Polyhedra, Artificial Intelligence, 4, pp. 95-119, 1973.

(Shortliffe 1976) E.H. Shortliffe, Computer-Based medical Consultation : MYCIN, American Elsevier, 1976.

(Shoup 1980) J.E. Shoup, Phonological Aspects of Speech recognition, in Trends in Speech Recognition, W. Lea, editor, Prentice-Hall, 1980.

(Waterman 1978) J. Waterman and F. Hayes-Roth, editors, Pattern-Directed Inference Systems, Academic Press, 1978.

(Woods 1976) W.A. Woods et al., Speech Understanding Systems, Final Technical Progress Report, Report n° 3438, vol. I-V., BBN, 1976.

(Zucker 1976) S.W. Zucker, Relaxation Labeling and the Reduction of Local Ambiguities, in Pattern Recognition and Artificial Intelligence, C.H. Chen editor, Academic Press, 1976.

(Zue 1979) V. Zue and R. Cole, Experiments in Spectrogram Reading, Proc. IEEE ICASSP, Washington, 1979.

Office Systems and Information Systems

CLARENCE A. ELLIS
Stanford University and Xerox Corporation

1. INTRODUCTION

Today's office is a changing environment. The days of manual typewriters, adding machines, and hand sorting, copying, and mailing are being replaced by integrated electronic office systems containing mixed media document editors, electronic mail systems, electronic files, and numerous personal electronic aids such as electronic spreadsheets and tickler files. Studies have shown that well over half of today's office workers use some form of information processing or telecommunications workstation on the job. The need for these systems is apparent: increasing administrative overhead and costs, coupled with increasing need for information and information processing are causing many organizations to search for methods to increase *efficiency* (amount of work getting done per dollar expended), and *effectiveness* (extent to which work done actually meets goals and needs of the organization). Organizations are finding that careful design and implementation of office information systems, taking into account important human factors of organizational design and office psychology and sociology, meet these needs, and can make some aspects of office work more pleasant and convenient.

Initial emphasis of the office automation thrust was concerned with clerical work and structured tasks such as repetitive forms fillout. A prime example of technological assistance for this class of work is the word processor, a computer based office aid which embodies the "typewriter with memory" notion with features to make the composition and editing of documents easy. The primary target of this machine was the secretary-typist. Only 6% of the office dollars are spent on secretary-typist salaries, and only 20% of these people's time is consumed by typing [77]. The word processing market is never the less a huge market. Two other categories are identified within the office work force. These are the "professional and technical workers" and the "managers and administrators". The combination of these two occupational categories is designated as "knowledge workers." These people typically spend a lot of their time performing work classified as unstructured tasks. An example of unstructured work is useful for clarification. A sales manager, when confronted with a threat of resignation because of discrimination by the company's best sales person, may avert this catastrophe by a combination of investigation, critical bargaining, and persuasive communications. Note that this involves creative problem solving for a one time exceptional situation rather than following a routine procedure. Henry Mintzberg, who conducted one of the most thorough investigations of managerial work, found that managers spend 66-80% of their time in oral communications [59]. Other studies have verified this. One study reported that knowledge workers, on the average, spend their time approximately as follows:

- formal meetings - 20% to 30%
- writing - 10%
- phone conversations - 5% to 15%
- reading - 8%
- analysis - 8%
- travel - 10%

The remaining time, classified as miscellaneous and idle includes activities such as informal meetings, expediting, scheduling, waiting in airports, seeking people or information, copying, filing, and transcribing. Given this mix of structured and unstructured work, it seems that the emphasis within office information systems for this latter category of workers should be on using the system as an augmenter and assistant rather than as a replacement for people doing repetitive procedures.

Today's advanced workstations have gone a long way toward making the office environment more productive, and the computer interface more friendly. A misspelled word in a document can be automatically detected and easily replaced without retyping the entire page or document. A user interface consisting of pictorial (iconic) objects such as pictures of file cabinets and mail boxes, displayed on a high resolution screen, and voice input / output is easy to learn, use and remember [74]. All of this is augmentation for the individual. It appears that the next big productivity breakthrough may occur when we can achieve community augmentation which works as well as systems being introduced today work to augment the individual. In the sales manager example mentioned above, note that many of the resources needed to solve the problem were non-local, such as access to shared historical information; ability to quickly communicate with many people; shared problem solving capability; knowledge concerning default procedures which typically are outside this manager's domain, and knowledge of formal and informal organizational structures. Later in this book, we will further discuss this important notion of communal office augmentation. Proponents of this view also stress the need for links between the augmented office and external systems, including the paper based operations, including telephones and databases, and including management information systems and decision support systems. Certainly, a prerequisite for the emergence of this type of augmentation is the integration of the electronic components of the office to allow them to function together smoothly and effectively. I believe that artificial intelligence and knowledge based systems are an important ingredient in this next big step.

As previously mentioned, office automation was primarily concerned with automating the clerical functions within the office. On the other hand, *management information systems* have traditionally been concerned with data processing aids for managers, where data access systems or data monitoring systems would periodically generate reports. For top executives and senior managers, *decision support systems* were developed as interactive real-time systems to provide the top decision makers with rapid access to data and to models which would summarize and extrapolate from the raw data. These tools range from simple cross tabulation to online models which estimate the cash flow of an organization under varying business strategies or varying economic climate assumptions. Recently there has been a realization that there is a large need for diverse information access and problem solving aids at all levels of the office. This is true because the office handles those aspects of the business which were not amenable to easy automation within the data processing department. Thus, office

information systems try to integrate and share information between all three of the segments (clerical worker, professional, and manager) in a manner which benefits problem solvers at all levels of the corporate structure.

There are two primary technological innovations which form the basis for this change in the office. One is advanced workstations, e.g. personal computers and word processors driven by the inexpensive microprocessor boom; and the other is advanced communications systems, e.g. new technological developments in the areas of PBX, local area networks, and satellites. This overview attempts to provide a description of there various components and tools within a modern office information system in the context of a coherent framework. This overview also addresses office modeling, office technologies, and the building blocks that help form an integrated office information system. Throughout these discussions, it will be assumed that readers have a basic understanding of computer hardware and software, but not necessarily of specialized sub-areas.

2. PEOPLE IN OFFICES

The office is not so much a place as a locus of common activity. Many people are following trends of making their home their office, utilizing the many available modes of electronic communication. Many office tasks are routinely performed in remote locations (by salespersons on airplanes for example) not at all resembling the stereotypical room with office desks and furniture. For these modes of work to be feasible, the social structures and organizational structures of the work organization must be explicitly recognized and nurtured. Potential social conflict can arise in the division of time between family and work when working at home. Potential organizational conflict can arise when a manager cannot accurately calibrate and evaluate the effectiveness of his/her subordinates because the subordinates are too frequently traveling. The office can be viewed as a conglomerate of people, information sources, and information manipulation tools which are drawn together by common goals and structures. We must recognize the presence and the importance of social structures and organizational structures in the office, and be sure that introduction of automated aids does not put these factors into negative imbalance.

At the level of individual and social needs, work within the field of social psychology provides useful information for office design. Work by Maslow has become famous because it elucidates layers of need within people [53]. This theory holds that there are five layers of needs. At layer 1 are some very basic *physiological needs* such as the need for food. One can inevitably observe the hunger drive which results if this need is not fulfilled. Above this layer, there are *safety needs*. Even is prehistoric times, the caveman represented an example of this in their relentless striving for a shelter safe from the animals of the wild. A third layer is called the *belonging need*. Because man is innately a social creature, there is a drive to find a group to which the individual can, at least to some extent, belong. This may be a family, an ethnic group or an organization. Above these layers which are called hygienic layers are two further layers called motivators. These layers pinpoint the need for *self esteem* (layer 4) and the need for *self actualization* (layer 5). In our modern western society, office workers tend to be above the threshold level for the hygienic layers, so Hertzberg found in his studies that an increase in these had no noticeable effect on the worker's productivity or effectiveness. On the other hand, his studies found that increase or decrease in motivators had significant effects.

At the organizational level, Professor Ouchi and others have shown that successful organizations have organizational control structures which fit to the size, style, and needs of the organization [64]. One prevalent structure is the *organizational hierarchy*, also referred to as the bureaucracy. In this structure, there are levels of management, and rules concerning what employees should do and how employees at each level should interact with the levels above and below them. For a large organization this allows delegation of work via standard channels. However, this structure can have a large amount of overhead from middle level managers, and from the amount of time and energy expended to get information from the top level to the actual workers at the bottom. Another organizational structure which is associated with Japanese business structure is the *clan*. Within this structure, employees within an organization feel that they are part of a family, and the employer is looked upon as the benevolent parent. There is a lot of company loyalty, and a lot of behavior is determined by peer group pressure. This has the advantage of a lot of devoted work being done without the need for a large bureaucratic overhead of managers to oversee or to pressure workers to get the work done. It has disadvantages of stifling individuality and not encouraging creative inventiveness. A third organizational structure is called the *marketplace*. In the marketplace the workers interact according to resource needs and resource availability. Clients or buyers search in the marketplace for contractors or sellers to perform the tasks or to provide the goods that are needed. If I need paper supplies, I would get them from the supplier which can provide them to me at the lowest price within the timeframe I need. Similarly, if I must deal with one of the accountants in the organization, I will choose to deal with the one who will charge the lowest price for services, but who will give me good service in a short waiting time. The market structure tends to be fair in its evaluation and rewards provided that the notions of cost of information can be concretely established. Although clients have some prior knowledge of the various contractors and capabilities, they must generally communicate with a number of potential contractors in order to determine the specific capabilities and current availability of the contractors. This tends to add a great deal of overhead.. Notice that a pure market structure for a corporation is not practical, but the availability of specially tailored electronic mail facilities makes some aspects, like the cost of bargaining and keeping informed, more feasible [83]. Thus this organizational structure and the two preceding ones illustrate the pure ideals whereas any actual organization would utilize a mixture of the above.

Given these observations concerning social and organizational structures, we next look at technology within the office. Afterwards, we can then put these ingredients together to suggest how technologies can and should fit into office structures.

3. TECHNOLOGY IN OFFICES

In this section, we briefly describe some of the technological innovations and systems which are typically included within integrated office information systems. Workstations, document editors, spreadsheets, networks, electronic mail, conferencing systems, and database management systems are a few of the components. Others not discussed here include electronic diaries, calendars, tickler files, smart cards, FAX, OCR, and TELETEX.

3.1. Workstations

Workstations are computer based systems which today typically sit on top of a worker's desk. Others are portable, including the briefcase computer, and the handheld computer. Many have an auxiliary disk which may be a floppy disk holding thousands of bytes, or a rigid disk holding millions of bytes. Many workstations have a screen to display documents, (text and graphics) sometimes in color. It may be a touch screen so that items on the screen can be selected by pointing with your finger, or it may have a mouse or other pointing device. Movement of the pointing device controls the movement of a cursor visible on the screen, and controls the selection of objects on the screen. Other facilities may include input output connections to devices such as a local printer, or to a network or PBX. The workstation may be integrated with the telephone, and provide enhanced telephony facilities such as phone number lookup and autodial and timed recall. The user interface to the workstation is an important aspect, and may include, for example, voice input and output. Window packages are also available which divide the screen up into rectangular regions called windows. These present a user interface that supports and explicitly makes visible the concept of multitasking and the execution of parallel processes. Each window can be associated with a process, and each can be independently created, sized (windows can usually overlap), started, interrupted, stopped, and destroyed by the user. Thus, while my computer program is compiling in one window, I can be reading my electronic mail in a second window, and writing a paper in a third.

3.2. Document Editors

Document editors have become more powerful and elaborate than simple word processors. They share with word processors the ability to

create, edit, and store electronically letters, forms, and reports. But they also may have the ability to create and manipulate other types of objects within documents such as graphics, voice, scanned in images and facsimile machine output. Full page screen editors have been replacing the older line editors. Some editors present a WYSIWYG image on the screen. WYSIWYG stands for What You See Is What You Get, and means that the image on the display screen is identical or very similar to the output that one would get at the printer, so no extra codes or editing marks are visible, and so that margins, tabs, and justifications are immediately visible. Some editors are more properly called document processors, because they include facilities for creating, structuring, spell checking and formatting documents. Sometimes these editors manipulate documents as structured objects, so that a document is a combination of paragraphs, sections, frames, and boxes, which in turn may be composed of other objects such as sentences or curved lines, or tables. These editors provide facilities for manipulation of the document structure as well as the document content.

3.3. Spreadsheets

Spreadsheets are two dimensional tables partially or fully displayed on screen based workstations whose entries can be specified as functions of other entries in the same or different tables. After the rules of dependency for each table entry are specified, the alteration of a particular entry immediately results in the updating of all other entries which depend upon the altered entry. Manual spreadsheets were used for many functions long before display based computerized spreadsheets were constructed. This is a good example of technological improvement (the low cost display based workstation) making feasible computer based tools (the computerized spreadsheet). Spreadsheets are useful for numerous calculations such as tax calculations, depreciation, standard deviations, and net present value calculations.

3.4. Networks

Local area networks and PBXs (standing for private branch exchanges) have been instrumental in allowing workstations to communicate with each other and with various mainframes and services. Transmission media over which this information may be transmitted include wire cable, coaxial cable, microwave, communication satellites, and optical fiber. These allow all workstations in a building or a floor of a building, or in some cases a campus to be interconnected. These can in turn be connected via long haul networks such as phone lines or satellite, to other branches of the same organization or other organizations. These long haul connections may connect distant parts of the country, or connect far away countries.

3.5. Electronic Mail

Message systems (also called electronic mail) and teleconferencing are asynchronous facilities allowing messages and documents to be sent over computer networks, usually delivered in a matter of seconds or minutes. The fact that these are asynchronous means that the recipient of the message does not need to be present at the time the message is sent. The next time that the recipient signs on to his or her workstation, s/he will be notified that messages are waiting to be read. Modern systems allow arbitrary documents which may contain graphics, voice, etc, to be mailed. Numerous features such as priority mail, distribution lists, and automatic forwarding are also available within modern systems. Computer based teleconferencing systems differ from mail systems in that an ordered transcript is kept of all contributions to the conference, and the default is a many to many conversation rather than a one to one conversation. Modern mail systems and conferencing systems are feature rich enough to be used for either one to one mailing or for conferencing. The term teleconferencing is also sometimes used to denote a synchronous distributed session where all parties must be simultaneously present at their workstations. By contrast, audio and video teleconferencing are synchronous conferencing systems in which dedicated transmission facilities connect special conference rooms for real time meetings. All of these technologies are destined to partially replace some face to face meetings, although the initial reception has not been as overwhelming as predicted.

3.6. Databases

Many workstations have personal databases as part of their local environment. The presence of communication facilities means that access to larger shared databases is also feasible. The advent of reliable efficient distributed databases means that the personal database can communicate with or merge with the shared communal database. Thus conferencing sessions or mail which a community of users may be interested in saving can be stored and retrieved on many keys such as date, topic, author, etc. Similarly, large collections of documents can be stored retrieved and searched very naturally if they are stored within a database facility. Databases appear to be an excellent base upon which many of the higher layer components of an office information system can be structured. This is particularly true if the database has efficient facilities for handling triggers, alerters, and varying record structures containing nonstandard data types such as image and voice [18,29,33].

4. A VIEW OF THE OFFICE

Given the social and organizational considerations of section 2 and the technological considerations of section 3, we can formulate a view of the office which takes into account both of these considerations. This will also pinpoint the unique aspects of office information systems. Some prominent features of offices and office systems which distinguish them from other (e.g. data processing) systems are:

- People Systems - An office is a social environment to which any introduction of procedural changes, goal changes, or automated equipment causes perturbations. If people who need to exchange information are moved so that they are no longer close to the coffee machine, then needed information, which used to get transferred in this informal way, may no longer get exchanged. Explicit consideration should be given beforehand to analyzing likely effects of changes. Many technologically successful systems have failed due to ignorance of human and social factors. For example, the mass movement of secretaries away from individual managers to word processing pools violated social maxims. The managers tended to feel a loss of a most valued resource, and the secretary tended to feel less directed and less motivated. In many cases, this change broke important social ties which had helped to keep many office organizations healthy.

- Dynamic Systems - Change is frequent and expected in most domains within the office. An employee's vacation days, for instance, force others to change their routines accordingly. Change also results from promotions, employee turnover, competition's changes, sickness, changing government regulations, etc.

- Concurrent Systems - The office is a highly parallel, highly asynchronous system. In many cases, this structure has grown from years of experiential learning by doing. There is much to be learned by systematically observing the informational and social checks and balances of a smoothly working organization. But also, one may observe habits and impediments which are unnecessary relics of a past way of doing business. Thus, some of our mathematical office analyses which automatically detect potential parallelism have produced extremely useful results. Inherent in the media of the electronic office is the much greater potential for parallel processing than the previous age of unautomated offices. As a word of caution, we have observed that it is not always easy to discern whether an activity falls into the category of unnecessary relic or necessary redundant checks and balances

- Ill-Structured Systems - In terms of monetary investment, we have mentioned that most of the people resource is within the "knowledge worker" category. These are the professionals, the managers, and the executives who are highly paid, and therefore are a prime set of candidates for office augmentation aids. Much of the work performed by this group is unstructured or ill-structured. These workers need augmenters and aids rather than the structured data processing systems which are prevalent in more structured parts of a business. For example, a sales manager in a typical company may need to search diverse data, read between the lines of a report, and have a confidential lunch meeting with a colleague in order to track down the information necessary to salvage the account of a big customer. A useful office information system should be able to assist this person in all of these aspects of work.

- Open-Ended Systems - Another important group of people is the "clerical/secretarial" category. One might assume that the work of

people in this category is all structured, but office studies have shown that even within this category, the amount of problem solving, exception handling, and customer interfacing (all three are unstructured activities) are high. Just the activity of interpreting a customer's handwriting may involve significant problem solving. Thus systems and models must be capable of handling a diverse spectrum of activities with high proportions of semi-structured and unstructured activities. We must insist that systems be inherently open-ended with escape hatches to handle unanticipated exceptions and emergencies. This is much more tenable than aiming toward notions of total automation of procedures, total removal of paper from the office, or totally peopleless offices.

5. OFFICE MODELS

Office models can be used by many different persons in many different ways. We cannot avoid the use of models; each worker in the office carries an informal model of his/her activities. Furthermore, there is high utility in explicitly devising and recognizing models within the office domain. Formal models, which tend to be mathematical and explicit, are useful for analysis and design; informal models, which tend to be implicit and ad hoc, are useful for capturing social and behavioral aspects. Before presenting an office information systems model in section 1.5 which will be our framework for discussion throughout this book, we will first attempt to answer some fundamental questions of what is modeling and why build models.

5.1. What is a Model

Models are limited abstractions of reality. The limitation is expressed by focusing upon a subset of the attributes and structures. Informal models can help workers to decide upon appropriate actions when they encounter unforeseen or exceptional situations, [76]. Explicit awareness of user models can help to improve upon the acceptance and error-free use of office systems. For example, whereas a computer person has a computer model of disk storage which implies that retrieval of a page of information does not destroy the original copy on the disk, an office worker (non-computer person) has a file cabinet image which says that when a page of information is taken out of a file, it obviously is no longer in the file. A user-friendly system design may choose to somehow take this into account; for example, there may be a DESTRUCTIVE READOUT command, and a NONDESTRUCTIVE READOUT. Formal models may range from social interaction models of matrix management to queuing and simulation models of bottlenecks and throughput in the flow of transactions within standard office procedures at peak hours. Both formal and informal models can be extremely valuable aids in the process of office systems design.

5.2. Why Modeling

Models, using the power of abstraction, frequently provide insight into important characteristics of complex systems. Simple "back of the envelope" models can be quite useful for first order approximations. Certain telephone operations models ignore much switching complexity, and give good estimates of number of lines which one can expect a PBX to be able to service. Generally speaking, mathematical models are used to provide rigorous mathematically tractable characterizations of, or approximations to, office systems. These models frequently provide insights into important characteristics of complex systems. One reason for this is because models can frequently be built at a high enough level of abstraction that properties, generalizations, individual differences and inconsistencies can often be discovered which would not otherwise be apparent. This argument in favor of modeling is based upon the thesis that insights obtained by modeling lead to understanding and theories, which lead to consistent and well-structured implementations. In addition to models having a potentially strong impact on systems design and implementation, practical data input and existing implementations provide good foundations, motivation and direction for the theoretician who is concocting a model. Indeed, some models demand that collected data or observations from existing implementations (or simulations) be used to drive the model (e.g. trace driven models, stochastic and statistical models). The aforementioned benefits of modeling must be traded off against the amount of time and effort needed for the model development. Modeling done hastily, haphazardly, and imprecisely has been known to produce misleading and in some cases grossly incorrect results. In the design of static, well-understood systems, modeling may not be cost effective. In considering the development of office information systems there are compelling arguments in favor of modeling: (1) the technology of the systems is still in the formative stage; (2) these systems are quite dynamic (changes to office procedures, office personnel or office requirements are frequent) and, (3) there is no comprehensive theory of office information systems. Indeed, there is strong reason to believe that the office of the future will need to lean heavily on modeling and analysis. Both qualitative and quantitative analysis can be performed using mathematical models. In this section we define a family of models which will be used formally and informally throughout this book in a variety of office situations to aid understanding and to perform analysis. In this way it is hoped that the value of modeling becomes apparent to readers. Furthermore, we firmly believe that data gathering, simulation, and measurement - evaluation are pragmatic activities which go hand - in - hand with mathematical modeling. Thus these activities should be tightly coupled with the theoretical modeling effort. In summary, the design of a system as complex and dynamic as an office information system can benefit from insights and theories developed through studies of office models.

5.3. Types of Models

In a previous section we defined an office model as a limited abstraction. This reflects our feelings that no one model spans all of our modeling needs, and that it is not possible to completely capture all aspects and nuances of a complex office situation. Factors of psychological make - up of office workers, acoustics, and even desk size may be very important to a particular office's success, and may not all be taken into account by a single model. Just capturing within a model all of the exceptional conditions which may occur may be an endless task. Thus it is useful to classify models and explore the utility of different types of models. One classification [63] of office models is based upon different possible views of the office adopted by the modeler which in turn may be based upon the modeling objectives and the modelers' disciplinary specialty:

- Information Flow Models. These models seek to represent office work in terms of units of information (forms, memos, etc.) that flow between offices; they allow us to define, generally by some type of network diagram, the operations performed on each information unit. They are useful in defining the types of information units involved in office work, and the range of operations applied to each unit.

- Procedural Models. This class of models is based on the view that office work is basically procedural in nature, involving the execution by office workers of pre-defined sequences of steps, called procedures. Like information flow models, procedural models involve operations (procedural steps) and operands (units of information). They emphasize the task-oriented nature of office work, in the sense that each procedure is designed to perform a particular task, and they identify some of the important and often unpredictable roles played by people in carrying out procedures. In these two respects they offer a more accurate model than information flow. However they may be less accurate in portraying inter-procedure and inter-departmental flows and dependencies.

- Decision-making Models. These models relate to the decision-making activities of managers and other office personnel. The more traditional model of decision-making treats it as a fairly objective process of information gathering and analysis; recent studies have suggested that there is a large element of unpredictability in organizational decision-making, which may be difficult to capture with this model.

- Database Models. Office work can be modeled in terms of databases; these contain information records that are created and manipulated by means of transactions, and that can be viewed by generating reports of the databases contents. Business accounting and control methods are nowadays largely based on this model.

- Behavioral Models. It is possible to view office work as a social activity, involving situations and encounters into which are woven the information-processing tasks of the office. Sociological studies provide a considerable amount of information to support this approach to office

modeling. Quality of working life models [38] are an interesting example of behavioral models.

• Organizational Models. We previously mentioned the importance of explicit representation of the formal and informal organizational structures. Conrath, Bair, and others [8,21] have validly argued the thesis that one of the most important and time consuming activities within the office is communication among people. Thus models of organizational communication have been applied to the office to study, among other things, who communicates most heavily and most frequently with whom.

These categories are not mutually exclusive or all inclusive; rather they show that modeling of offices can be approached from different conceptual viewpoints.

5.4. Uses of Models

It can be debated whether current simplistic models of offices are adequate for study purposes. It appears that this depends upon the use which is to be made of the model. There are a number of ways in which models are inherently inadequate (particularly modeling human behavior), but also there are significant ways in which office models have proved themselves useful:

- Description and specification of offices
- To assist in office systems design and implementation
- Dynamic simulation of office activity
- Evaluation and comparison of systems and equipment
- Prediction of system performance under changing conditions
- As an aid to communication among designers and analysts
- As an aid to understanding by workers, managers or other stakeholders.

6. OFFICE FRAMEWORK

Previous discussion of facilities and tools such as document editors, spreadsheets, and electronic mail was rather random. It suggested that we need a design model that is comprehensive enough to serve as a framework for discussion. The model around which we will structure much of the discussion in this section is called the "Interconnect Model." It is a layered framework for discussing the pieces of an office information system and for explaining how these pieces fit together. One part of the model is concerned with the structuring of facilities and tools within a personal environment, and the other part of the model is concerned with structuring within a communal environment. For each of these environments we postulate that functional work gets done via an implicit or explicit hierarchy of processes. A diagram of this framework is shown in figure 1.

The layering concept, as we will use it within the interconnect model, is a technique to structure complex systems. It has been used successfully to aid in the conceptualization and in the construction of complex systems of many types. A good example of the use of layering is within the international standards organization's OSI model of communication protocols [87] which we adopt as one component of our model. This model has seven layers ranging from a wires and bits description at layer one to application protocol descriptions at layer seven. In general, a given layer implements a set of functions for use by the layer above the given layer. Each function is constructed from combinations of more primitive functions provided by the layer below the given layer. Note that there is simplification in a well structured system, because a given layer only needs to be aware of and deal with the layer immediately below it. We define interfaces to be the functions, procedures, and data structures of a given layer which are known and usable by the next higher layer. Note that here is another opportunity for simplification, because a given layer can implement information hiding, and only make available needed procedures and data structures to the next higher layer, rather than all of its algorithms and complexity. For example, when I am driving my car from home to work, I really make use of layered functionality. I turn the steering wheel to the left when I want the car to turn to the left; and I turn the steering wheel to the right when I want the car to turn to the right. I do not know or particularly care exactly how the electro-mechanical subsystem carries out this function as long as it works. This electro-mechanical subsystem represents a lower layer which provides the driver with car turning functionality while hiding the details of mechanical leverage and wheel torque suspension.

Another important concept is concerned with protocols. We define protocols to be agreed upon sequences of actions by several entities (such as people or workstations) which they perform whenever they want to exchange information at the same layer in the hierarchy. For example, when answering the phone, it is customary to say a polite greeting such as "Hello." This "Hello" is not passing critical conversational information, but it is a useful action to let the caller know that a connection is complete. In this sense, the "Hello" is a protocol. This example illustrates that protocols can be useful to gain a common understanding for communication without much overhead Without protocols, mass confusion could ensue. To understand how protocols and multiple layers fit together, consider a three layer example of offline conferencing within a multinational corporation. If the chief executive officer wants to make a statement to all of the executive officers, then he dictates this message in his native tongue (say German, if he is located in Germany) to his executive scribe who performs translation. When speaking, the chief executive officer considers that he is speaking directly to other executive officers using mutually understood protocols which we will call level 3 protocols. In reality, he passes his message to his scribe at layer 2 across the 3/2 interface. The layer 2 scribe has a task of using agreed upon protocols to communicate with all other scribes using layer 2 protocols. For example, this scribe may translate the German message into English as the agreed upon common language among scribes. In reality, the scribe implements his task by passing the English message to the courier at layer 1 across the 2/1 interface. The actual means of transmission may be computer network, telegram, personal delivery, or other means depending upon the layer 1 protocol.When the courier working for the receiving executive officer receives the message (irrespective of what language it is in), he delivers it to his translator. If the receiving officer is based in Japan, then the translator may convert the text to Japanese and then pass it up to the officer at the top level. Note that the protocol at any layer can be changed without affecting other layers; this is great for system maintainability and alterability. Note that the only real transmission of information among different locations of the corporation takes place at the lowest layer. Since all communications at layer 3 must pass through all lower layers and upon receipt, pass up all layers, one potential implementation disadvantage of this structuring scheme has to do with the overhead incurred by frequently moving down and up the hierarchy. The reader should bear in mind that the framework which we are introducing is for the purpose of conceptual modularization, and says nothing about how layers are actually implemented. Thus implementations may choose to shortcut some layers of the system.

It has been postulated that individuals get work done via a hierarchy of processes [17]. For example, a typical office task such as making a telephone call is accomplished by activities at many layers. At the cognitive layer, we must recall the phone number and fix in mind the goal of the communication (i.e. what we want to say). At the motor layer, our fingers must dial the correct digits and our vocal chords must formulate the utterances which compose the spoken conversation. Thus we again encounter layers. Also when a person is interacting with a workstation, there are high level activities such as conceptualizing the layout of a document, and low level activities such as keyboarding. Both are required for successful interaction. This is also true on the receiving end to interpret stimuli from the personal environment. My low level sensory perceptrons notice such things as the alteration by the personal environment of the pixels on my display screen. Within my internal processing system, this may cause a signal and pictorial information to be sent up through various layers to a cognitive layer (my brain) which realizes that these bits form a new window which has just opened. At a higher semantic layer, I interpret this stimulus to mean that a high priority message has arrived at my workstation which I should read immediately. Although it is not the primary focus of this book, we note in passing that this user hierarchy which interacts with the workstation hierarchy should not be ignored in systems

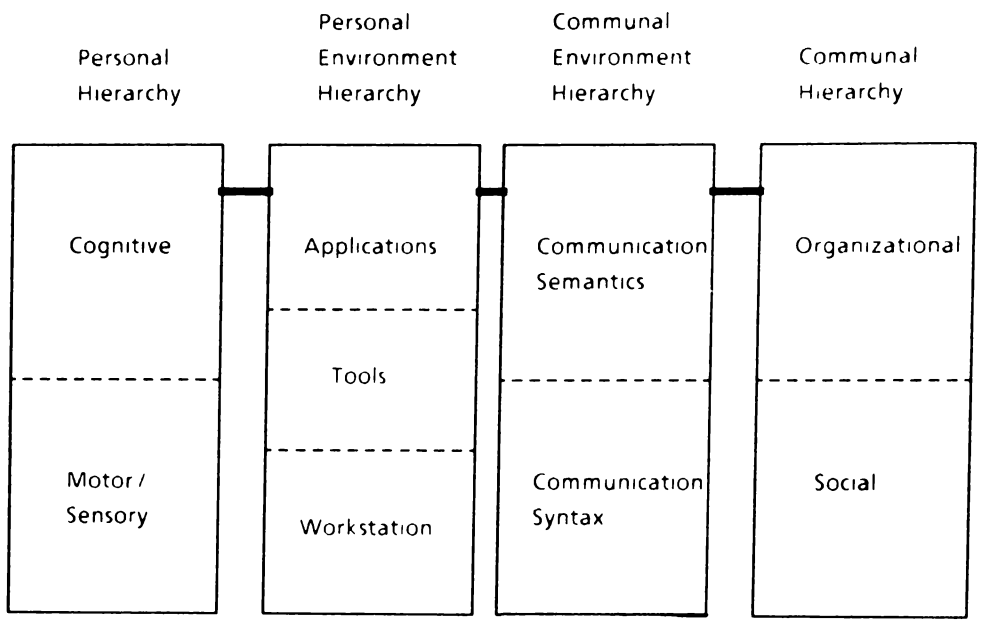


FIGURE 1

The Interconnect Model

design. Likewise, in our discussion of the communal environment hierarchy, we must remember that the social and organizational hierarchies should not be ignored in system design. All of these hierarchies are present within the interconnect model. (see figure 1)

It is possible to divide the facilities provided by the personal environment into three layers, each of which has two sublayers as illustrated in figure 2. There is the primitives layer, the tools layer, and the applications layer. Within the primitives layer, there is a basic workstation layer that provides facilities that are considered ordinary fundamental hardware and software. CPU, main memory, and "ordinary" input - output facilities are included here. An example of "ordinary" hardware in this category is the standard keyboard. The next higher sublayer in this hierarchy, called layer 1b in figure 2, is the structured workstation layer. Above this layer users see a system whose memory is very large, and potentially different in size and nature from the actual main memory of the layer 1 raw machine. This layer may include other special instructions, special input or output facilities, and generally special hardware for special applications. Note that in all of the layering specification, the exact boundaries between layers is hazy, and subject to change as primitives migrate from layer to layer. For example, color graphics which can be considered "special" today, can be expected to be more generally adopted as standard hardware in the future when technological progress sufficiently lowers the incremental cost of color.. The particular time and setting may be partial determinants of whether any particular feature is considered special or ordinary. Layer 2a is the basic tools layer which contains fundamental facilities useful by a wide variety of applications. Examples are compilers and window packages. Similarly, the facilities provided by the structured tools layer are application independent and used for a wide variety of applications, but they are typically more sophistocated tools usually implemented using or on top of the layer 2a tools. Examples are document editors and database facilities. These are typically built using a compiled language and they may occupy and run inside of a window, so we place them within layer 2b. At layer 3a are basic applications packages such as the spreadsheet packages and the forms design systems. These are generic applications which are applicable to many domains of application. The structured applications layer (layer 3b) contains domain specific application packages and systems. Examples include financial planning systems, and income tax calculation packages. There are facilities for which it is unclear in which layer they should reside such as calculators and calendars. Also, the exact nature, implementation, and sophistocation of a package may determine in which layer it resides. For example a database management system may reside at level 2b as a tool, but if a graphical forms user interface is added or if it is customized for a particular order entry department with a very high level interface, then it should be moved into the applications layer. All of these layers will be explored in more detail in chapter 2.

For layering of the various communal aids, we use the communications layering developed and standardized by the international standards organization (ISO model, [87]). This has seven layers as shown in figure 3. The lowest layer is the physical layer which is concerned with transmission of raw bit streams. The datalink layer (layer 2) is concerned with the reliable transmission of frames of data. The next layer is the network layer and is concerned with point to point routing of datagrams, and the avoidance of congestion. The fourth layer is the transport layer which is concerned with reliable host to host communication for use by the session layer. The session layer is concerned with setting up and maintaining and tearing down process to process connections. The presentation layer (layer 6) performs transformations on the data to be sent, such as text compression. The seventh and highest layer is the applications layer which contains protocols for specific applications so that generally available services can be communicated with and used by any cooperating customer. This includes generic applications such as file transfer protocols, and very domain specific protocols such as those used by the banking industry. Whereas the lower layers are typically covered in books and courses concerning communications, the higher layers receive much less study and are much less developed. The area of office information systems design is intimately concerned and interwoven with the higher layers of the personal and communal environments.

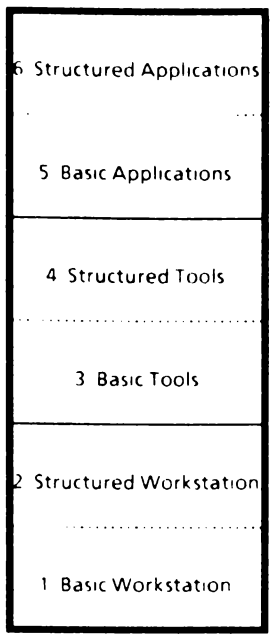


FIGURE 2

The Interconnect Model

(personal environment)

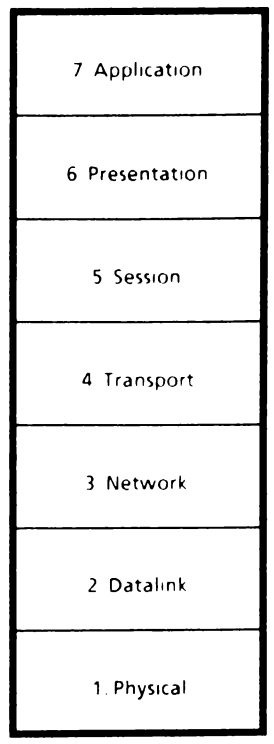


FIGURE 3

The Interconnect Model

(communal environment)

7. WORKSTATION ORGANIZATION

The workstation is a facility (or station) to aid in getting work done. As such, it may include a desk, chair, paper, pencils, lamp, and coffee cup. To narrow this to our domain of study which is modern office information systems, we concentrate on those components which provide a computing environment for the end user. Components which we consider include processors, memories, input output facilities, and peripherals within distributed and centralized configurations. In this section, we discuss possible configurations, and try to define and categorize workstations and their subsystems. Then we present a technical introduction to some workstation peripherals which are particularly relevant in an office environment - secondary memory systems, video displays, keyboards, pointing devices, and printers. Advanced primitives and office tools are then discussed in later sections.

In this section we will define terms such as intelligent terminal, word processor, and personal computer, and compare these with our workstation concept. In order to do this we will first elaborate on our office information system architectural philosophy. A large number of distributed and centralized configurations can fit into the general 3 component office information system architecture that we propose here.

The first component of our architecture is local computing power, which is derived from the observation that computing power (the processor) once was relatively expensive, and now is quite inexpensive. Thus it is sensible today to have powerful computing capabilities, and storage capabilities also, as local resources within one's workstation. Thus we will be addressing the processor subsystem of a workstation later in this section.

Our office information system architectural philosophy has a second component based upon the observation that offices are social structures, and that there is a fundamental need for people in the office to commmunicate. *Communication facilities and protocols* are thus an important component. This emphasis implies that a workstation should have IO ports with hardware and high bandwidth to communicate with high speed networks. There is also a need for capability of real time transfer of non-standard data types such as voice, color graphics, pictorial images, or animated information displays, all of which require large amounts of data to be represented succinctly and transmitted rapidly. Higher level tools and applications built upon the basic communication networks and protocols must reflect workstyles and methods actually in practice within the office. Sharing of information, bargaining, and communal problem solving can be supported by shared files and distributed databases, and by phone communication aids such as auto-dialers, answering machines, and call retry facilities.

Realizing that there has always been a need to share resources within the office, our office information system architectural philosophy has a third component which is the notion of *servers* which provide *services*. A server is a controller of one or more shared resources. A client i.e. user, workstation, or another server, can request use of this resource from the given server, and it is the server's function to provide the service - that is the sharing or the resource - on some equitable or prioritized basis. An example of a server which can be made available via a communication network is the file server which provides service of remote storage and retrieval of files utilizing some mass storage medium.

A decade ago, the cost of computing power (a processor) was one thousand times higher than today, and it frequently made economic sense to share the processor by means of a computation server, which is the well known batch or timesharing service. Because the cost curve has changed drastically with the flourishing of minicomputers and microcomputers, tasks are done on the workstation that a decade age would have needed to be sent to the central computer. Although the economics of this particular server are quite different today, our architectural philosophy suggests that given any particular stage in technological development, there always will be a class of server type resources which are expensive or bulky or have other characteristics which make them desirably provided on a remote shared basis. One type of resource which falls into this class is the service explicitly involved in communications or multiple person interaction. For example the mail server should probably be disjoint from the local workstations, and can usefully be distributed on multiple servers. This implies that if one mail accepting server or its node of the network is broken or unavailable at the time I want to send a message, then another server at a different node can accept, store, and forward my

message. These servers must obviously communicate and coordinate among themselves to implement a reliable and consistent service. Thus requirements of performance and reliability can sometimes suggest that a distributed service is appropriate.

Within the architectural philosophy described above, there are a spectrum of architectures, from those based upon dumb terminals (teletypewriter like devices having no processor and essentially no memory), up to stand-alone personal computers, which have computation and memory facilities, but no network connection capability. With a personal computer, all of the facilities at one's disposal are local; with a teletype, all of the facilities are remote. The name glass teletype is given to graphical display units which closely mimic a teletype. Once information is output, it cannot be altered. A glass teletype, like a teletype, presents a user interface which allows only appending of text to what is displayed, not arbitrary updating. These devices also lose information when it is scrolled off the screen. Some interesting points in the spectrum are the intelligent terminal, the cluster configuration, and the networked workstation. Consider, for example, the act of editing a 20 page document to correct the spelling of a word. In a dumb terminal system, all editing actions and storage occur at a central computer called a mainframe computer which is usually configured to serve many terminals simultaneously. The dumb terminal contains a small buffer which can hold a few characters, so each character printed is sent to the mainframe, and then sent back by the mainframe in order to be displayed on the screen or typewriter output device. This is called remote echoing, and requires less local capability than local echoing. In a cluster configuration, there is a computer processor and memory system to service a group of terminals that are located in close proximity to each other and the controller. These computers, called cluster controllers, may also be connected to each other via a communication network. Intelligent terminals are terminals with added memory, and processing capability. Although an intelligent terminal may or may not have enough memory to store 20 pages of document locally, it can buffer some amount of text; at least a line and frequently a screen full or several pages. This is adequate for some local editing to take place. Thus the correction of the word, as mentioned earlier, would be done locally upon a buffered copy of some portion of the document. At some later time, this buffer of altered text would be stored back onto the cluster controller as part of the updated document. In the case of a standalone word processor or personal computer, all of the storage and processing of a 20 page document is local; this is also the case with modern networked workstations - which are becoming ever more powerful and capable.

We can list four characteristics that an office workstation should possess:

- local computation power,
- communication among users,
- access to remote services, and
- powerful, friendly user interface.

Note that the dumb terminal and the standalone personal computer are forms of workstations that are weak in some of these areas. Thus we expect these workstations to be replaced and displaced as the cost performance ratio continues to decrease for sophisticated workstations.

7.1. Processor Subsystem

The computing power of a fairly large 1971 computer cost $50,000 in that day (which was also the year of the birth of the microprocessor) and the same computing power can be obtained within a microcomputer on a chip today for under $50. The pioneering INTEL 4004, which introduced the era of the microprocessor in 1971, was a 4 bit CPU containing approximately 2250 transistors which had a repertoire of 45 executable instructions. The microprocessors within microcomputer based workstations have grown from 8 bit processors to 16 bit processors to 32 bit processors, which were introduced in 1972, 1974, and 1981 respectively. The larger word sizes generally imply more processing power and addressing capability. As an example today, the Hewlett-Packard HP9000 workstation is fabricated from a 32 bit microprocessor which contains approximately 450,000 transistors and offers a repertoire of 230 instructions. Near future trends show a continued increase in density and decrease in size and cost of processors, because of the application of very large scale integration (VLSI) for the fabrication of chips. The trend is away from bit-slice processors and toward multiprocessor microcomputers having multiple single chip processors. The bit-slice processor is a processor built by the concatenation of smaller word size components. For example, a 16 bit register within a CPU could be built from four 4-bit devices, such as the well known Advanced Micro Devices' AMD 2901 and AMD 2903. Each year, the complexity of chips seems to double and the cost per function falls by about 25%. This means that in the longer term future, it is probable that intelligent workstation type devices will be used by the billions. Estimates predict that before the year 2000, the population of microprocessors will exceed the population of people living on this planet. The challenge is to structure and utilize this technology in constructive, humanizing ways.

7.2. Memory Subsystem

The memory subsystem provides both primary and secondary memory. By secondary memory, we mean those memory resources beyond the main memory storage system. This may include local hard disk, local floppy disks, and remote file servers or optical disks arranged in a hierarchy, all of which we refer to as secondary storage. By primary memory, we include main memory, and other auxiliary rapid access memories smaller than disk such as caches. Typical primary memory sizes range from 64K bytes to 8M bytes; secondary storage will be discussed later in this section. The physical and economic limitations on size of main memory imply a need for virtual memory capabilities for many office applications. Systems handling large text files, graphical data, color, voice, and scanned in images are factors in the office system which require virtual memory schemes, which in turn require fast local storage and / or high bandwidth access to nonlocal storage. Office workstations are now available with dedicated memory management units to handle the paging and address translation mechanisms. Some workstations have all of their secondary storage remote, and access this via high speed network. Another factor influencing memory subsystem design is reliability. Error detection and correction codes are being widely employed, and "smart" memory controllers are being used to detect and isolate segments of memory subject to a relatively large number of errors.

7.3. Input Output Subsystem

The input output subsystem has a requirement to address three areas: interfacing to the network, interconnection of devices which are integral to the workstation (such as display), and the interconnection of other peripheral devices to the workstation. If the environment is network based, then this places constraints on the design of the input out (i/o) system, because there typically is a minimum acceptable response time for communication over the network. Transparent access to files across the network means that reaponse time should closely match that provided by the bandwidth of the comunication subsystem and of primary memory. Thus the network interface might require a DMA channel for direct full-bandwidth access to primary memory. We mentioned that the display or CRT is a commonly used interface which we consider integral to the workstation. This graphics subsystem places a heavy demand on the i/o subsystem; some workstation configurations provide a separate communications facility for the display in order to increase the bandwidth available to other devices. Another strategy used sometimes is to provide one communications facility for high speed devices, and a separate facility with its own controller for low speed devices. In any case the trend is toward more functionality and better user interfaces supported by the local workstation hardware.

8. WORKSTATION PERIPHERALS

When we discuss peripherals in this section, such as secondary storage technology, display and keyboard technology, pointing and drawing devices, and printers, our discussions are predicated on the assumption that many of these peripherals could be implemented as nonlocal servers or as an integral part of the basic workstation. We have already mentioned secondary memory and the graphical display unit. Consider the local printer sitting on my desk, which prints slowly but accurately and is really only used by myself. This facility could alternatively be provided by a shared nonlocal printer which might receive requests from many individuals. This is the notion of a print server which accepts documents transmitted via a communications network and prints them. I probably must walk to a separate server room to pick up my output, but the printer may be faster and produce higher quality. At

any point in time, elements of need, performance expectation, technology, and economic situation all help to dictate whether a particular installation will implement certain functionality as a shared remote service or as a non-shared workstation local capability. One office may purchase a small printer attached to each of its workstations, while another office that has need to print large volumes of high quality lengthy documents with color graphics may choose to buy a single large and fast laser printer and make it a shared resource available to all corporate workstations. Some offices may implement both of the above.

8.1. Secondary Storage Technology

Storage systems have traditionally been organized as storage layers. So one speaks of primary storage (main memory), secondary storage (frequently disk or tape), tertiary storage (frequently remote backup), etc. For classical mainframe systems, secondary storage of large tapes, drums, and disks have been prevalent. We thus find that many large installations today have information quickly available using large disk systems such as the IBM 3350 which holds 317 megabytes (MB) per spindle with an average access time of 25 milliseconds (msec), or the even larger IBM 3380 which holds 1260 MB per spindle and average access of 16 msec. In this discussion our attention will be focused upon secondary and backup storage for office systems, so much emphasis will be on physically small facilities. With the advent of personal computers and office computers, other size / price ranges of disks have become very popular. As a rule of thumb, main memory holds thousands of bytes (or low millions), disk holds millions of bytes (floppy disks in the low millions, winchester disks in the high millions), and optical disks hold billions of bytes (or low trillions). We will next take up a short discussion of floppy disks, winchester disks, and optical disks. Other technologies such as digital tape cassettes, cartridges, and non-winchester disks, though less prevalent, can also be found as secondary storage in some office systems.

The *floppy disk*, more properly called the flexible disk, consists of a flexible thin sheet of plastic material with a magnetic coating and grooves arranged in concentric circles (called tracks). This somewhat resembles a small bendable phonograph record, and the *disk drive* is similar to a record player with its arm and head and motor. When the disk is placed inside of the drive and the drive is turned on, the disk spins at around xxx revolutions per minute. The placing of the head, which reads and writes digital binary data, upon the correct track is controlled by software.The floppy disk is removable, so it can be carried in a briefcase or if it is a smaller disk, in a shirt pocket. The floppy disk originated in 1970 from IBM as a backup storage medium for information backup and recovery purposes. It has since gained much popularity with personal computers as secondary storage because hard disks have been too expensive and tapes, as a sequential storage medium, have been too slow.

The need within offices has always been to make equipment in and around your desk as small as possible. So floppy disks are smaller than the typical 14 inch hard disks, and over the years floppy disks have been getting smaller. Many systems originally used 8" disks, then 5.5" minidisks, and then 3" to 3.5" microdisks. Early floppy disks held under 85 thousand bytes (abbreviate KB) of data; today most floppy disks hold hundreds of KB with newer technologies allowing over 20MB of data storage on a single platter. Disk drives for these floppy disks vary in their dimensions. Many are around 8" deep and 2" to 4" high. But drives have also shrunk to meet demand, so that half height drives are available (1" to 1.7" is typical height) for 8" and for smaller disks. Users must beware that floppy disks are not all standardized, so sizes vary, drive interfaces vary, and capacity and speeds vary.Each groove is a track of digital storage capability, and disks may have from 20 to several hundred tracks per inch. Disks may be single or double density, and single or double sided. This can result in storage density up to xxx, and speeds in the range of xxx to xxx.

The *Winchester disk* is a sealed rigid magnetic oxide medium disk which typically holds 10 to 50 MB of data, and typically has a diameter of 8", 5.25", or 3 (up to 3.5)". The Winchester architecture means that the disk can sit much closer to the disk than a non - Winchester disk. It also means that dust, and other contaminations that might happen to a floppy disk are minimized since the disk is sealed . Most Winchester disks are not removable from their drive. Compared to floppy disks, Winchesters are generally faster, higher capacity, and more expensive. Note that the same miniaturization that happened with floppy disks has occurred with Winchesters; they have shrunk from 8" to under 4". The sub 4" Winchesters are particularly convenient for offices, and many of them are designed to withstand the rigors of the portable computer environment. Typical Winchester characteristics are exhibited by the IBM PC-AT disk.

It is apparent that further media advances will continue to decrease the size and increase the capacity of Winchester disks. Winchester disks capable of storing 600 MB of data are now being produced. Many manufacturers are using *plated media*, especially for smaller diameter disk drives. These thin film media have higher coercivity and rigidity than oxide media, which imply respectively higher capacity and more ruggedness. Another type of medium useful for Winchesters is *stretched surface recording* which uses flexible magnetic disk material stretched over a plastic substrate. Proponents of this medium say that this taut but resilient surface combines excellent recording characteristics with high resistance to damage from contaminants. Dozens of other explorations are in progress in many areas of Winchester design; simultaneously many developments in non - Winchester technologies are also in progress and may have an strong influence on the future of Winchester disks. At the low capacity end of the scale, we mentioned that the technology of primary storage (RAM) can now produce million byte memories. As cost per bit continually decreases, RAM cartridges with integral batteries to refresh the memory may provide a MB or more of fast, compact nonvolatile memory.

At the high capacity end of the scale, *optical disks* hold high promise of replacing magnetic media in the farther future. Today one can store many gegabytes of data on a single 12" optical disk surface. Though today's single optical disks can store larger quantities of data than magnetic disk or tape, most of them are not erasable. Characteristics include high density, relatively slow rotation speed, removability of disks from drive, and as mentioned, non-erasability. These non-erasable optical disks are actually a new class of mass storage peripherals.

The *write once optical disks* represent a relatively fail safe medium for permanent archival storage. They differ from magnetic tape as computer archives because they are quasi - random access, need no rewinding, and have a longer lifetime. Some manufacturers guarantee their disks for ten years or more. Still tape is valuable as an alterable archive for relatively short term storage when speed of access is not critical. Some of these write once optical disk systems are huge and expensive, holding 128 disks and accessing ten to the thirteenth bits of data in less than five seconds.

The *read only optical disks* are a convenient medium to inexpensively mass produce and disseminate large unchanging or periodically updated bodies of information such as software and databases. These disks can be stamped out for well under $10 apiece. Many of these read only optical disk systems are medium capacity, storing one or a few gegabytes with average access time around 200 msec. Some disks use glass or plastic sandwiched around the sensitive recording layer. This keeps dust and dirt away from the recording surface. Since its optical, the read mechanism can be much farther away from the disk than magnetic technologies. Because of this distance between the outer surface and the recording layer, any dust or dirt on the outer layer is out of focus and thus non-interfering. The techniques involved in recording data onto optical disks vary greatly. One technique used is the *bubble forming process* in which the write laser beam heats up the medium forming gas that is trapped under a metal substrate but has enough force to deform the covering layer. The read beam of the laser in these systems detects the deformity of the surface, and signals a binary one. Another recording technique is called the *oblative process* in which the write process melts a hole in a tellurium surface via laser. These holes are then detected by their changed reflectivity in the presence of the read laser to distinguish a one from a zero.

Some *erasable optical disks* are built upon this ablative writing process By mixing sub-metals with the tellurium, it is possible to erase by using phase changes in the impure tellurium. Other erasable disks use magneto-optic techniques which also support the erasure operation. The vast capacity of optical disks is allowing entire large software systems to be recorded and distributed or mailed. Shipment is typically not a problem for these plastic, cartridge encased read-only disks. Also, write once disks allow reliable logs and archival store in permanent

fashion. For example, whereas a single write once disk could be altered without detection by burning more holes, a system which redundantly writes a disk and its complement (switched ones and zeros) provides a reliable double disk record where alteration is always detectable. Large jukebox write once systems are being tested for document storage in the United States Library of Congress. Finally, erasable storage combines the flexibility of Winchesters, the removability of floppy disks, and the economies of scale of optics. Although significant engineering obstacles remain, optical technology is very promising for the future.

8.2. Video Display Technology

Our means of interaction with computers has changed over the last decades. Punched cards and paper tape in the 1960s gave way to the printing terminal used with time sharing systems of the 1970s. These terminals were characterized by clumsy editing of text, limited browsing, and almost no graphics. Introduction of the *cathode ray tube* (CRT) terminal changed this and today's office system frequently is based upon workstations which are display based for immediate feedback, and editable output. CRT technology is a flexible one which, prior to office popularity, has already been investigated and mass produced for television. It is capable of providing an interactive document composition and editing medium which allows fluid editing, fast scrolling, graphic arts, plotting of information, and spatial information handling including photographic image handling. In this subsection we describe some of the most important parameters of video displays, and several approaches to CRT design: character generator CRT, bit-map CRT, vector display CRT, and storage tube CRT. We also describe some display technologies which are alternatives to the CRT.

There are three basic components needed for cathode ray tube display: the CRT itself, the refresh memory, and the display processor. In the most common applications, as television display, an electron beam inside the CRT is directed to a large number of light emitting spots on the inside of the front surface of the CRT. A light emitting coating on the inside of this surface, called the *phosphor*, causes visible illumination when the electron beam strikes any of the points on the screen. The particular properties of the phosphor determine the visible screen properties such as color, persistence, and efficiency. The electron beam sweeps out a series of horizontal lines, called *rasters*, as it moves down the screen. Changes in beam intensity as it moves along cause changes in the brightness of the screen image at various points. In this way, the screen image is built line by line, but this process of displaying the image on the screen, called refreshing the screen, must be repeated frequently because the spot of light dies out rapidly (within a fraction of a thousandth of a second). The process of making a pass over the screen to display the image is called a *frame*, and the time that it takes to refresh the screen, typically one fiftieth or one sixtieth of a second, is called the refresh rate. Television CRTs typically have an image consisting of 525 rasters (about 350 of them visible) of 424 dots (called pixels as an abbreviation for "picture elements") each. With interlaced displays, every other one of these rasters is refreshed during a frame, and the remainder are refreshed during the next frame. This works well for moving pictures on televisions, but with stationary images (particularly narrow horizontal lines) it can lead to *flicker*. This is a psychophysical phenomenon , also called flicker fusion, whereby most people notice the individual images flashing on and off if images are displayed at a rate of less than 30 frames per second. When it is presented at a rate of 60 images per second or more, the images seem to "fuse" and most people see one continuous image. *Bandwidth* is a measure of how fast the CRT's electron beam can be turned on and off, which is closely related to the potential number of dots per raster, which is closely related to the number of characters per line. If the bandwidth of a CRT is 4 megahertz, this is equivalent to about eight on-off transitions per microsecond, and each horizontal sweep of the electron beam takes 53 microseconds, then 424 pixels can be displayed in a row. If the characters are six dots wide (including one dot to separate characters), then 70 of them can be displayed on a line; if they are eight dots wide (as is commonly the case) then only 53 will fit. It also follows that non-interlaced displays operating at standard video rates can display only about half the number of rasters that the interlaced can (i.e., about 175 lines). For characters 8 dots tall, (the minimum for legibility), this implies a capacity of 22 lines of characters. This is reduced to 17 lines if the characters are a more reasonable 10 dots tall (including inter-line spacing). Several additional factors enter the calculation when color displays are considered. Color displays typically have three electron beams all moving across their rasters together for red, blue, and green, or mixtures thereof. With specially constructed monitors, the bandwidth can be increased to thereby increase the number of rasters, and/or dots per raster. However the cost of these is usually much higher. Color monitors are available with 1000 dots per line, but at great expense.

The refresh memory of a CRT display is used to store the data indicating which screen dots should be on and which off, and this information must be quickly accessible since it is used during each refresh cycle. One memory arrangement used in many of today's office CRT systems is to have one bit of the refresh memory correspond to each dot location on the screen. The memory is accessed sequentially, synchronously with the scanning of the electron beam. If a "1" bit is found in a given memory location then the beam is turned on for an instant, creating a dot in the corresponding screen location: if the bit is a "0" then the beam is left off. A CRT display with this arrangement is known as a *bit-map display*, because there is a one to one correspondence between the bits of the refresh memory and the dot locations on the screen. This approach is used in graphic arts terminals, and in the high-end office workstations, but not for low cost workstations and terminals. The representation of every screen dot of a 2000 character display with 80 pixels per character requires a 160,000 bit memory. Thus, although the electonic design of a basic bit map display is very simple, it is a relatively expensive approach because of the cost (although this is decreasing) of the large amount of memory which must be used.

An alternative to the bit map is to divide the screen dots into fixed size rectangular character frames and use the display memory to store codes for the sequence of characters to be displayed rather than to store actual bits. This technique is used in the vast majority of the less expensive display based workstations such as the proliferous dumb terminal. This typically reduces the number of bits per character which must be stored in the display memory from 80 (or at the minimum 48) to 8 (or 7). The tradeoff for the savings of memory is that additional display processing hardware is needed to convert the code (ASCII is frequently used) into a pattern of dots during the refresh cycle. The dot producing hardware is called a character generator and this type of display is called a *character generator CRT*. It sometimes works by real time table lookup (in a ROM) of the bits corresponding to codes in the display memory. Thus although a character may appear many places on the screen, its bits are only stored once in the ROM lookup table. The input to the character generator consists of the code of the character to be displayed next and the current raster number. The generator then looks up the corresponding five or seven bits in the ROM table, and passes these to the electron beam circuitry, which in turn produces dots on the screen. All of the bits of a single raster are processed before the next row is begun, and this must be repeated 7 or 9 times, depending upon the height of the characters. Note that the use of character codes, fixed size character frames, and a character generator precludes variable spacing and sizing of characters; generalized graphics capabilities are also precluded.

In our attempt to succinctly cover general technology and methods, we do not do full justice to many innovative imbellishments of these basic CRT designs to overcome many of the above mentioned problems. For example, extra bits are frequently added to the character codes in the refresh memory to specify that the character should be bold, or italic, or blinking, or etc. By having the character lookup table in RAM rather than ROM, it is easy to provide tools to edit character looks. Similarly with bit map systems, character looks and international character sets can be easily constructed and edited. Furthermore, pixel is a more general term than bit or dot, so that grey level, color, and other embellishments can be added to bit map displays. Some of these systems have 24 bits per pixel; this adds up to a lot of memory for the display memory.

In applications like computer assisted engineering design and circuit board layout, the type of display most often used is a *vector display CRT*. The vector display draws and continually refreshes lines (vectors) on the screen. It differs from the raster scan display in that its electron beam can move in any direction rather than being confined to a fixed pattern of scanning rasters from top to bottom. The vector display outputs two endpoint values to a vector generation circuit which directs the beam to draw the line connecting the two points on the screen. The

accuracy of the vector display is usually better than the raster display because the latter can only produce points on the raster while the format canposition them arbitrarily. Lines generated by a raster which are not horizontal or vertical may have a stair step appearance. This effect, called the "jaggies" is not present in vectors on a vector display. Characters are represented on the screen by a series of vectors, a "v" might be 2 vectors, a "w" four, and curved characters such as "s" may consist of many vectors. Flicker sometimes occurs with vector displays when the number of vectors becomes too large (usually in the tens of thousands). This is because the refresh process must draw each vector sequentially, and time is consumed by this process. Vector displays can be further refined in their handling of the display memory to provide instantaneous sizing and dragging and continuous scrolling and zooming. This instantaneous response is possible, but harder to attain with raster displays. As a result, vector terminals are popular in areas such as the composition of display ads. On the other hand, this type of display may exhibit weaknesses of limited capacity, inability to display solid areas and photographic imagery, and minimal color capacity.

A third type of CRT that is used in some office systems is the *storage tube*. The storage tube may draw images either as rasters or as vectors (usually as vectors). Once an image is drawn, the spots which are turned on remain on due to a constant flooding of the screen with background electrons which are only strong enough to cause any region of the phosphor which is excited (i.e. illuminated) to remain so. Thus, refreshing and a refresh memory are not necessary. However, images cannot be moved or deleted without erasing and regenerating the entire screen. This can be a slow operation for complex images, and also requires that the entire screen image be stored somewhere. This negates the primary storage tube advantage - that memory is not required. Thus the storage tube is mostly used in applications where updating is relatively infrequent, and where adding to an image is much more frequent than moving or deleting.

Flat panel technologies are becoming increasingly important due to the importance of small workstations and portables. In the remainder of this subsection, we briefly discuss technologies which can produce flat display screens. *Light emitting diodes*, abbreviated LEDs, are unidirectional diode-like elements which can be driven by the low voltages of common digital circuits. These displays are commonly found on calculators, watches and digital clocks. They are also available in red, green, and blue, so full color displays can be fabricated. Single line LED displays are also becoming common on electronic typewriters and portable computers. Although large arrays of LEDs have been produced, the cost of fabrication is high and the power consumption is extremely high. It is this very high power consumption which may limit the future use of this technology.

Liquid crystal displays, abbreviated LCDs, are also commonly associated with calculators and watches. The liquid crystal display elements produce no light themselves, but can be switched between two states, one which permits light to pass through, and one which does not. These elements in conjunction with back lighting or reflected lighting can be easily fabricated into arrays using techniques similar to those for integrated circuit production. Liquid crystals share with LEDs the advantage of low voltage operation, but they also offer the additional advantage of low power consumption. Although there have been some reports of production difficulties, large LCD arrays have been used for games, for portable televisions, and for portable terminals. LCDs are relatively slow in their response time (a tenth of a second to turn a dot on or off), so serious smearing during scrolling or copying of blocks can occur. Nevertheless, they are compact, light weight, and consume far less power than CRTs.

Vacuum flourescent displays are another possible contender in the flat panel display field. This device is analogous to the CRT in that electrons travel from a cathode through a vacuum and strike a phosphor. But in the case of the vacuum flourescent (VF) display, the entire back surface of the device is the cathode. On the inside surface of the front glass is an array of independently selectable transparent anodes, each covered with phosphor. When a particular anode is turned on, it attracts electrons from the cathode. These strike the phosphor, causing a bright spot. This technology has the drawback of requiring high voltages, and the power consumption is comparable to a CRT. On the other hand, the manufacturing process is straightforward and can readily accommodate special requirements such as color, size, etc. Panels based on this technology are available commercially in sizes up to 256 by 256 dots, but the cost of these panels makes them uncompetitive with CRTs in most cases.

Plasma panels, another technology for flat displays, may be thought of as thousands of tiny neon tubes sandwiched between two glass plates, one carrying vertical rows of electrodes, the other carrying horizontal ones. When the proper signal is applied to a given horizontal and a given vertical electrode, the dot at the intersection of the two electrodes lights. Once lit, the element can be maintained in the "on" state by a voltage far lower than that required to turn it on. This property means that given the proper driving circuits, plasma panels have a built-in memory. Dots that are lit stay lit without further attention. In this respect, the plasma panel is like the storage tube. However, unlike the storage tube, elements can be selectively turned off as well. As disadvantages, the plasma panel has a relatively slow response time (tenth of a second), and high voltages are required by the driving electronics (too high to be handled by ordinary integrated circuits) and high power consumption. The interconnection problem which plasma panels share with other flat panel technologies is also a difficult one. This technology, like the VF display, requires the development of high voltage integrated circuits to become really competitive.

Electroluminescent displays are based on substances which give off a glow when current passes through them. LEDs do this, but only in a restricted, specially fabricated region. Electroluminescent displays, (EL panels) have material which glows all over. For many years, the common use of electroluminescence has been in flat panel night lights. This technology shares the problems of high voltage requirements, and high power consumption. Nevertheless, EL panels of up to 240 by 320 dots are available on the market.

The above paragraphs outlined the basics of some display technologies which are already on the (rapidly changing) market. Other technologies are available or in research labs that were not mentioned. Advances in integrated circuit technology are bringing the costs down, so many non-CRT displays are becoming competitive with the CRT.

8.3. Keyboard Technology

The keyboard has made a transition from typewritier to integral built-in component of the computer, to separate workstation peripheral. Today, many keyboards come from a factory in their own cases; they are ready to plug into a computer or monitor and contain their own printed circuit boards. Because it is almost the only component of a workstation that isn't shrinking, a keyboard is the logical place to put an interface for a mouse, a voice recognition device or an optical character reader. This independence of the keyboard is particularly noticable in the cordless keyboards available today.

The keyboard industry seems to be technology driven, and also ergonomics driven. On the technology side, the most important force in the 1970s was the introduction of *capacitance key switching* to replace *magnetic switching*. Capacitance switching depends on the change in capacity between two charged plates or pads that occurs when a key is depressed. This technology quickly outpaced more expensive, magnet-based technologies, including Hall-effect, inductive core and reed switching - all still used in applications requiring keyboards with particularly long lifetimes. For applications requiring flat or minimum travel keys which seem not to move when pressed, *membrane switching* has been used. In this technology, the flat keypad switch is simply two layers of polyester film traced with silver conductor, separated by a spacer sheet that has holes in contact areas. When the key is pressed, the layers make contact activating the circuit connection. A conventional key set can also press down the top membrane, so that a high duty data entry keyboard can result. Another technology which has been especially popular in Japan is conductive rubber switching. Keyswitches are produced by molding a sheet of conductive rubber so it forms a dome under each key. When a key is pressed, the dome collapses to close a circuit and then pops up again. On the ergonomic side, each of these latter two technologies frequently produce keyboards which seem to have an uncomfortable and unnatural feel. The full travel capacitance keyboards "give the feel people are used to and are reliable enough to last the lifetime of the product." Flat panel keyboards, although simple to make and low cost, frequently give no tactile feedback and thus are uncomfortable for high volume typists.

The conductive rubber keyboard, on the other hand frequently requires an excessive amount of force to depress the keys. "It doesn't have the electric typewriter feel that secretaries are used to," is a frequent comment. This notion of comfortable feel is sometimes determined by market presence and available equipment rather than optimality in ease of use. A de facto standard has been brought about by the popularity of the IBM PC and the large number of people who have grown accustomed to its keyboard feel. Thus many smaller manufacturers are following the IBM PC keyboard specifications.

Ergonomics is concerned with the comfort, speed, and naturalness of the devices used by workers. If a person in the office will use a keyboard for large numbers of hours per day, it is worthwhile expending effort to make the keyboard, and the other aspects of the workstation environment comfortable and as free as possible of detrimental physical or physiological factors. The height of the keyboard and of the display should be appropriate to the person using them. Thus, adjustable heights are good, and having the keyboard disconnected from the display allows it to be moved and positioned where it is convenient. The infrared (cordless) keyboard is one with which some users have become enamored because they can carry it some distance from the workstation (e.g. to an easy chair) and type with the keyboard in their lap. Although not the recommended posture for lengthy attentive work, it has its advantages. In a number of environments, such as construction and manufacturing facilities, users have a real need for remote keyboards. In high use environments, where it is particularly important to consider user comfort and safety, more and more offices and organizations are adopting equipment which conforms with the German Deutsche Industrie Norm (DIN). This German standard emerged in 1980 after much ergonomic research. For usage in Germany, it requires that keyboards have a low profile (the modern look) with slightly less up and down key movement than older, generally accepted keyboards, and sculpted keycaps with tactile feel. A maximum height of 30 mm from the top of the desk to the *home row* is required. The home row is the row upon which the fingers of a touch typist normally rest; this is the third row for most keyboards. The popular low profile keyboards introduced in 1975 were about twice that height. Key travel (vertical distance the key moves when pressed) set by DIN is 0.120" to 0.150" as compared to 0.150 to 0.190" as the pre-DIN norm. The specification by DIN of tactile instead of linear feel full travel keys is leading many companies to redesign their keyboards. German researchers found that this measurement and the angle of the key rows are crucial in preventing operator fatigue.

In terms of speed, the arrangement of keys which is called the qwerty keyboard is known to be far from optimal. The name qwerty is derived from the labels of the first six keys in the top alphabetic row of this type of keyboard. This keyboard layout was designed during the 19th century for typesetting purposes, and it was necessary that the keying in of data be maintained at a regular pace that would not exceed the machine's ability to punch paper tape which would later be fed into a hot metal linecaster. Therefore the qwerty keyboard was arranged with frequently used keys placed far apart from each other so that two of them would not be pressed in too close succession. Thus, our layout today is based upon an engineering design constructed to degrade the operator's keying speed. The "Guiness Book of Records" testifies to the effects of this keyboard design: a 1.4% increase in maximum typing speed over an 18 year period. In 1923, the world record on a commonly used keybar typewriter was 147 words per minute; in 1941, it was 149 wpm on a standard electric model. The highest speed recorded for five consecutive years after that required a different kind of keyboard, namely the Dvorak-Dealey "Simplified Arrangement." August Dvorak and William Dealey were not the first to attempt to break the hold of the qwerty design. Qwerty had found immediate acceptance, and as time went by, became increasingly difficult to dislodge from public favor. In 1944, a U.S. Department of Navy study substantiated the benefits of the new keyboard design. However, the cost of retraining operators and acquiring Dvorak keyboards was judged excessive, and thus the Navy rejected the Dvorak keyboard. In 1974, Lillian G. Malt and S. W. Hobday were granted a patent in the United Kingdom for the "Maltron" keyboard which was designed with a horizontal row curve to fit the unequal lengths of fingers. This thereby drastically decreased fatigue experienced by typists when forced to maintain an angle in forearm and wrist to keyboard in qwerty systems. Maltron keyboards have been accepted to some extent in Europe, often among people who had no prior typing training. Other schemes have been considered, and some are still being researched. Experience has been reported with keyboards upon which you slide your fingers without lifting to spell out the most common words, and with five or six key single hand sets where various chords are played to compose the various letters of the alphabet. In conclusion, we see that keyboard technology, and keyboard ergonomics are continuing areas of research and development and are interesting and significant aspects of office systems

8.4. Pointing and Drawing Devices

When working with a display based office workstation, you are pointing much of the time - at text (by moving the cursor), graphics, or a menu selection. In the section on user interfaces, we will discuss how designers attempt to make these activities easy and efficient. One aspect of this is the pointing device used. Pointing devices are available in many shapes and forms. In this subsection, we will discuss some of the prevalent ones, but our descriptions are not exhaustive.

- Cursor keys. In most display based text editing systems, an arrow, or some symbol is displayed on the screen to denote the current type-in position. And these systems, even if they have a pointing device with them, let you change the position of the symbol (called a cursor) using the keyboard. Some of these systems use control characters, and some have four special keys called cursor keys which respectively move the cursor one character to the left, right, up, and down. Cursor keys work well for moving a short distance because your hands do not need to leave the keyboard. However, they are not so convenient for moving long distances, so systems sometimes provide keyboard commands to move one word or one paragraph or move to the beginning of the current line or ... While effective, these commands complicate system operation. Finally, we mention that cursor keys are not useful for most graphics. They do not work for free form drawing, area selection, palete choice, or small movements of a pixel or two.

- Cursor disks and joydisks. Some word processors have a round flat area on the keyboard surface which you press to move the cursor. This is one effort to overcome the problem of moving arbitrary directions with arbitrary speed. The relative part of the cursor disk that you press determines the direction that the cursor will move. Thus if you touch the upper left side, the cursor will move toward the left and the top of the screen. Many of these disks are implemented via four dielectric areas. These are laid out so your finger produces a greater capacitance change toward the disk edge, causing the cursor to move faster. Similarly if you press harder, more of your finger contacts the pad increasing the capacitance and the cursor will move faster. The joydisk is a similar device. It is a rigid disk imbedded in the keyboard surface that can be pressed in any one of eight directions. Cursor speed can be increased by tricks such as simultaneously pressing the shift key. The speed of the cursor movement can be adjusted by software. Neither cursor disk nor joydisk works for free hand drawing or complex pointing.

- Digitizer tablets. For free form drawing and for engineering drawing such as computer aided design, high resolution digitizer tablets that work with electromagnetic arrays are used. You use the tablets by moving a stylus or crosshair over the surface. The tablet contains a two dimensional grid of fine wires, from 200 to 2000 wires per inch. Control electronics send electrical pulses down each wire in turn. A small electric coil in a stylus or crosshair base picks up an induced current as the pulses come near. When the pulse passes right underneath, the current changes direction and marks the location. These units are particularly well suited for precision drawing and transferring existing complex designs. Large units have as much as a 42" by 60" working area. With suitable software, some systems can convert handwritten input into ASCII characters. Other technologies of tablets are available including ultrasonic tablets and pantographic tablets. The cost of these tablets tends to be high, so they are mostly used for special purpose precision drawing.

- Touch pads. Much lower cost are touch pads which use the finger rather than a stylus to operate.These are excellent for education and other work with children, but tend to be too coarse in resolution and too erratic for professional uses. Koala offers a touch pad which employs two membranes coated with a resistive layer and oriented at right angles. When these touch, the two axes resistance gives the location. The output is two continuous analog signals. The four inch square

Koala pad has 256 by 256 points of resolution. A number of companies including NEC and Alps offer similar pads. The PowerPad by Chalk Board combines two sheets with parallel conducting strips ant right angles. When you press down, you close contacts between two strips. The resolution is 100 points per square inch over a 12 inch square pad. The quality and resolution of touch pads are increasing and the cost of digitizer tablets are decreasing, so that these may be increasingly suitable for use by professionals in the office. Touch pads seem particularly suitable for portable workstation; there are numerous features becoming available with this technology that we have not mentioned that look particularly promising.

• Touch screens. These offer the most natural action of all: you simply point with your finger at something on the screen. Touch sentive screens adopt many technologies to locate your finger. Hewlett Packard's model 150 microcomputer puts an array of infrared light emitting diodes and photo transistor detectors around the screen. The system resolves 160 locations. There are 1920 characters or 199,680 pixels on the screen. Touch Technology uses glass that is coated with conductive indium tin oxide. Its system senses the capacitance change when a finger touches the coating The screen positions can be either fixed, with 32 or 64 discrete coating patches, or relative, where two bus bvarsxxx sense the relative capacitance anywhere over a complettrexxx coated screen surface. Resolution in this system is limited to approximately 100 by 100. Still other touch systems use two flexible transparent plastic sheets with parallel strips of conductive coating laid together at right angles. A finger pressing the two sheets together effectively closes a switch.

Touch screens have major drawbacks. Fingers are much larger than the characters on the screen. Touching the screen obscures what you are pointing at. Holding your finger up to the screen for long periods is tiring. And fingers leave prints on the screen. With adequate resolution, software tricks can eliminate some of the touch screen problems. Touch screens work well for novices, and in situations where you are simply passing through - at a store display for instance. Some users after the first few weeks find that a more efficient, less simple pointing mechanism is preferable. For example, menu selection which is often provided by pointing with the finger on touch screens can be done more quickly from the keyboard with a little experience.

• Light pens. These pen shaped devices, which function similarly to touch screens, allow fairly natural movement (although the screen angle is usually awkward). The pen contains a light receptor. When activated by pressing the pen against the CRT face, the receptor detects the scanning beam. A timing circuit compares the beam against the scanning raster and locates the pen's position. Theh pen works only on illuminated points. Light pens can point at much higher resolution than fingers, but share the arm fatigue problem. The pen itself requires a storage location and extra steps to pick up and put down. Although popular in the early years of computer aided design, light pens have been falling out of favor.

• Mouse. Although originally developed in the 1960s by Doug Engelbart's group at Stanford Research Institute, mice have reached the commercial market in quantity only within the past few years. Since your arm rests on the desk top, mice offer fairly natural movement with minimal fatigue. Designers have tried many different mouse variants, with no clear winners. The two principal types are mechanically driven balls and optically sensed grids. *Optical mice* offer quiet, reliable operation but require a special pad. *Mechanical ball mice* are cheaper but sometimes noisier. They may also require a pad if the desk surface is too smooth or too irregular. Perfect contact between ball and surface in not necessary since you watch a pointer on the screen while moving the mouse. Minor slippage does not disrupt operation. Most mice detect unlimited movement. Software that needs limited movement such as word processing programs, ignores movement beyond boundaries. Movement detection in optical mice is via a phototransistor which registers passing lines in the optical pad. Mechanical mice encode a turning shaft with optical sensors or mechanical switches. Detecting movement direction uses quadrature coding, a scheme that compares the output of two sensors for each axis of motion. Because of a small offset in the sensor location, the lead or lag on one sensor with respect to the other gives the direction.

As with pads and tablets, the desk space occupied by mice poses a major problem. Many people don't have enough clear space next to their computer. (But then a few years ago they did not have enough space for the computer itself.) Some computer furniture with fancy dropped cutouts for the keyboard may not work well with mice; other furniture has been designed with a convenient pull-out surface called "mousing space." To operate a mouse you must take your hand away from the keyboard, making mice less satifsfactory than keys for moving very small distances, such as one character at a time.

Despite the problems many people, including former skeptics, find mice the most satisfactory general purpose pointing devices. Movements of just a pixel are fairly easy, and if necessary the "gearing" between a mouse movement and the corresponding screen movement can be changed by software. Mice have many attractive features: A mouse stays where you leave it. And if you run out of space moving the mouse, you simply pick it up and roll it over the same area. Mice generally detect relative position. Mice can even serve (somewhat awkwardly) for free form drawing, since they can work over small areas as absolute positioning devices.

However, drawing with a mouse is less satisfactory than with a pencil shaped stylus because a mouse does not let you move with your fingers, only with your arm and wrist. To meet this problem, Ampower's Track Pen is essentially a repackaged mouse shaped like a fat pen. Its feel may make it more satisfactory than a mouse for extended design work with considerable drawing. To maintain north / south / east / west directions, the sculpted handle fits comfortably only in the correct orientation. Because you must pick it up and put it down with each use, the track pen is less convenient than a mouse for many routine applications. Instead of pressing a mouse button, you push down on the pen to activate a switch. If you need to move a mouse while holding down a button, the movement can be rather uncomfortable. The problem is that most mouse buttons are on top, so you must press down with your index finger while moving the mouse with your thumb and other fingers. A button that falls within a natural grip would work better, yet such buttons would get in the way of normal mouse movement when you do not want to press the button.

The buttons themselves have raised many arguments among mouse advocates. For simplicity, Apple has favored one button, which you operate in three distinct ways: A single click generally selects an item; a double click might select an item and perform an additional function, and holding down the button and moving the mouse (dragging) lets you draw arbitrary shapes. One button is too limiting say some mouse experts. Better to have more choices. No, reply the single button purists, you may have to look at the buttons to find them and remember what each does. Mice should be felt but not seen. As usual, the choice is a trade off. One button works best for beginners. Yet multiple buttons offer flexibility and can eliminate the need for dragging. Instead you can press one button to start and another to stop, and move the mouse in the interim comfortably. Many multiple button mice assign a single permanent function, such as calling up menus, to a second button. For a portable computer that you might operate on your lap, mice are rather inconvenient.

• Other pointing devices. Trackballs and joysticks are found mainly in computer games, and they work well for gross movement but not for the motions commonly needed with business and professional software. Trackballs are upside down ball-driven mice.They work well, without consuming desk space, for quick moves in one direction such as firing pretend lazers at attacking monsters, but much less well for precise two dimensional positioning as in drawing or word processing. No office computer system, to my knowledge, uses trackballs for pointing. Most joysticks' motion is coupled mechanically to a pair of potentiometers. Since the potentiometers have a limited range that corresponds to screen size, the joystick is generally an absolute position device. It too is possible but quite clumsy for office work. There are many exotic possibilities for future pointing devices. Some which are currently available in the office equipment market include the foot mouse, the eye tracker which moves a cursor directed by the motion of your eye, and voice activated systems.

8.5. Printer Technology

Much of office printing derives from improvement of typewriter output. Thus, emphasis is placed upon quality and speed relative to typewriter

output. Desk top printers are usually small, belonging to a single individual, and relatively slow (typically one or two hundred characters per minute). Shared print servers, on the other hand, may be fast, fancy, and expensive. Typical speed for a modern laser print server is several hundred pages per minute. Current estimates show that about 1650 different printers are commercially available, spanning more than 15 different printing technolo_ies. Thus, in this subsection, we will only give a brief overview of a sampling of prevalent technologies. To begin, we will define some commonly used terms that categorize printers.

- Printing Technology - impact printers versus non-impact printers

Impact printers are printers which use some variation of the standard typewriter printing mechanism where a hammer strikes paper through inked ribbon. Non-impact printers use chewmical, hehat (thermal), or electrical signals to etch or induce symbols onto paper. Many of these non-impact printers require special coated or treated paper.

- Character Form - fully formed characters versus dot matrix

Fully formed characters are constructed from solid lines and curves, like the characters of a typewriter. A dot matrix character is constructed from a carefully arranged sequence of dots packed very close together. Most solid character printers use an impact printing technology such as drum, band, chain / train or daisy wheel / thimble. Most dot matrix printers construct characters by printheads that fire solenoid driven wires, produce electrical discharges, create laser beam images, or use heat impulses.

- Printing Sequence - serial versus line versus page

These terms describe the amount of information that a printer can output in parallel or within a single cycle. Serial means that the printer proceeds across the page from right to left printing one character at a time. A line printer has mechanisms to print all of the characters of one line of a page simultaneously. Finally, a page printer can output a whole page of characters and images simultaneously or during one cycle which may take only a fraction of a second per page.

- Print Quality - draft versus correspondence quality versus letter quality

Print quality has traditionally been measured against office correspondence standards. As the name implies, draft quality output is OK for rough drafts, payrolls, and internal materials. It is readable, but not esthetically pleasing. Correspondence quality printing is good quality legible print that can be used for most common correspondence in busines applications. Closely packed dot matrix is sometimes acceptable for this level of quality However some business letters to important clients, potential clients, and special friends is meant to provide a quality impression. In these cases, letter quality is desirable: this means highly legible print resembling letters typed on a finest quality typewriter. This quality is becoming more and more expected and in demand for wordprocessing and formal business communications.

Next we mention sosme of the printing technologies in use within offices today.

The daisy wheel, or print wheel is a fully formed character serial impact printer using a metal or plastic print element with a circular series of flexible spokes rediating from a hub. The printer rotates the wheel to position the arm with the proper character, then hammerts the arm forward onto the ribbon and paper. Each spoke carries one embossed character, with 88 or 96 spokes per wheel, although some daisywheels have as many as 250 characters per interchangable wheel. Printwheel printers have long dominated the market for letter quality printers, but are now receiving challenges from other faster technologies. Typical speeds range from 20 characters per second to 100 characters per second, and are considered slow. Costs, in the low thousands of dollars are considered high. Recent advancements in the technology of printwheels include fiberglass construction for strength, cartridges to protect them, and refinements in motors to increase effieiency of the printing process and wheel life.

Dot matrix printers are the most prevalent printer for personal computing systems. They are very low cost relative to many other technologies; where else can an individual get a flexible reliable printer for under $300? This impact technology uses a printhead containing banks of wires fired at high speeds against an inked ribbon and paper. Wires are arranged in certain matrix formats such as 5 by 7 or 7 by 9. Firing is timed to produce a dot grouping (matrix) in the desired character shape. Both serial and line printers are available, and speeds range from 40 cps to over a thousand cps. A disadvantage of this technology is that print quality is low; however product trends such as multipass printing and overlapping printing among others are improving the print quality. Also improvements are being made in noise level, reliability, bidirectional printing, proportional spacing, and storage of numerous character sets in ROMs and PROMs.

Band and drum printers are the printers traditionally found in computer rooms. The drum printer is the parent of the band printer It uses characters embossed around a cylindrical drum which rotates on a horizontal axis for positioning. A hammer then strikes the back of the paper and ribbon onto the drum. Some of the disadvantages of this technology such as limited number of fonts, drum not easily changed by operator, and problems with print quality, are being remedied by band printers, sometimes called chain printers or train printers. Like the drum, this technology is an impact line printer with fully formed characters. It employs hammers that strike a rapidly rotating metal or plastic band of embossed characters into inked ribbon and paper. This mid-range mid-speed printer operates at a rate of thousands of lines per minute, and costs in the mid-thousands of dollars range. Although these have traditionally been noisy and bulky, vendors are making them lighter weight, more compact, adding sound proofing, acoustic cabinets, and are using modular construction and higher production volumes to drive down noise and price. Thus, band printers are appearing in more and more offices and remote processing server stations.

Ink jet printers use a variety of methods to spray a controlled stream of tiny ink droplets accurately onto paper, forming either dot matrix or solid characters. These printers are non-impact, and thus are relatively silent, which is a quality being requested by office users and demanded by regulatory agencies. Some ink jet printers use a simple low cost method for low speed printers called drop-on-demand. Ink flows from a closed reservoir or cartridge through a filter and into a multi-nozzle printhead. Ink is then shot toware the paper under control of an electrical pulse signal. Another method is called the continuous stream method. It uses a single nozzle head to eject an ink stream. Rapid horizontal head motion breaks the stream into single droplets which then are charged by an electrode. The charged drops are deflected vertically onto the paper by two other charged plates. The large percentage of the ink not used for character formation falls into a gutter to be recycled through the system. Ink jet printers range in speed from 50 cps to above 300 cps. This technology has been used well for production of color and of elaborate graphics.

Thermal printers are a relatively low cost, quiet, flexible technology which is best suited for limited small business usage. One can obtain a few hundred cps print speed for a few thousand dollars. The printhead in this technology moves across a specially treaated, heat sensitive paper. Individual wires in the head are selectively heated to darken the paper in dot matrix patterns or graphics. Distributed thermal heads use one heating element for several columns in a head that moves back and forth as characters are printed. Others use heating elements for every dot. Disadvantages of this technology center around the special heat sensitive paper: it's expensive, it's sometimes difficult to get, it doesn't resemble plain paper, and it fades with age. Work is continuing on improving these aspects

Electrostatic printers. This relatively fast printing technology (18,000 lines per minute) uses a specially coated paper that passes over matrix elements of styli, which then pass electrically charged dots in character groupings onto the paper. The paper then goes through a toner bath The charged dots attract ink particles to create character images. This has disadvantages that special paper and toner are expensive, and its not practical for preprinted forms. Print quality is usually less acceptable than xerographic process discussed next.

Laser / xerographic printers. This is a quality, high speed, high volume

technology which works in a non-impact fashion on plain paper or preprinted forms. One scheme uses a rotating polygon mirror to deflect a modulated fiberoptic light beam, or laser beam, onto the photosensitive surface of a drum or belt. This latent image attracts toner to the imaged areas. This toner is then electrostatically transferred to the paper and fused into a permanent image. Speeds can range from 10 pages per minute to above 215 pages per minute. Unfortunately, the excellent quality, speed, and flexibility of this technology are very expensive; not affordable as a personal printer, and currently too expensive for small installations as a server facility.

9. OFFICE ENVIRONMENT PRIMITIVES

This section presents some of the *advanced primitives* which are the basis for the implementation of workable office information systems. Workable encompasses such notions as responsiveness, user-friendliness, and extensibility.

In section 9.1 we discuss voice manipulation hardware and software, and in section 9.2 we discuss the graphical RasterOp instruction. These are examples of structured primitives (layer 1b of our model) which can transform the basic workstation into a more structured workstation for more efficient and convenient implementation of office systems functions. There are many other primitives at this layer such as mechanisms for efficient switching of contexts, for virtual memory management, and for system reliability and resilience.

9.1. Voice Technology

Some of the most interesting recent developments in office systems have been concerned with voice input and output. One can imagine that the sometimes clumsy keyboard and pointing interface could be replaced or augmented by a voice interface in which the user speaks to the workstation, and the workstation replies with answers or suggestions. In particular, there is promise that the transcription of dictation by a typist can be eliminated in some cases by computerized voice recognition. Also, there are some cases where it is advantageous for the computer output to be spoken rather than written, and this is possible today. In this subsection, we will explore voice generation via canned messages and via synthesis; we will explore voice recognition in the case of limited single utterances up to more ambitious cases of unlimited vocabulary; and we will explore voice editing which allows the speaker or receiver to insert, delete, and to perform other editing operations. Voice transmission, voice store and forward, and phone aids will be discussed in the communal aids section of this book.

Voice generation is the process by which a computer speaks to the user. One example of this is the telephone system which produces the message "We're sorry, but your phone call cannot be completed as dialed." This voice that we hear dispensing official telephone information has changed in recent years - it is frequently not human. This message that you hear can be stored and retrieved as a response to appropriate situations (called canned messages) or it can be pieced together from individual human utterances of syllables which are digitized, stored or burned into chips, and programmed to be combined in various ways on command to form words and sentences. Alternatively, the message can be derived by speech synthesis, also known as text to speech processing, which produces sounds that closely imitate human speech. Using signal generators and phonemic algorithms (programming instructions for the smallest particles of speech,) synthesizers create synthetic audio signals which approximate human voice in speed, pitch and inflection. Although speech synthesis is potentially more general and more storage efficient than digitized utterances , there is much which is still unkown about the human voice synthesis process. Speech synthesis chips are used, for example, in products which are programmed to answer the telephone when a user calls, to obtain text data from a host database in response to touch tone inputs, and to supply the text data in voice to the caller. A primary benefit of this technology is that it allows an ordinary touch tone telephone to become a data access terminal although this medium typically provides no means for scrutinizing and manipulating the output in non-realtime mode (see voice editing subsection).

Some speech synthesizers operate on text in ASCII or similar format. The synthesizer first compares each word to a dictionary of thousands of words that are exceptions to the rules of standard pronunciation. This dictionary can be expanded to include names and trade terms such as acronyms. Next the system applies letter-to-sound rules to determine proper pronunciation for words not found in the dictionary. There typically are hundreds of these rules. Next heuristics are applied to allow for the effect of surrounding words or characters on pronunciation. These heuristics can resolve many ambiguities such as "St. Frances St." where the first St. should be pronounced "saint" and the second St. should be pronounced "street." Finally, a digital signal processor creates waveforms using a computer model of the human voice tract to generate the actual speech sounds. In most cases, the synthesis process produces accurate speech. In other cases, the user must correct the system by entering the phonemic spelling of the word in the exceptions dictionary, typing the phonemic version of the word whenever using it, or typing a deliberate misspelling of the word that will pronounce correctly.

In the voice recognition arena, there are systems which recognize spoken commands (single isolated utterances), limited vocabulary sentences, and unlimited vocabulary. There are recognizers which are speaker dependent, and some which are speaker independent. Speech recognition is typically a pattern matching process. The recognizer captures a spoken word and matches it to reference words stored in memory. Speech enters a microphone as an analog signal, i.e. sound waves. The recognizer filters the sound into as many as 16 separate frequency bands and then digitizes the waveform in each band. These frequency bands contain information that we use when we hear and understand speech. Filtering eliminates a lot of garbage that has nothing to do with speech, such as noise. Digitizing the data breaks it down into pitch (frequency) and intensity (digital value). The digital values represent intensity levels within the frequency band. The human ear can perceive about 250 distinct intensity levels and about 1000 unique pitches. The recognition system accepts and stores this information in two modes. The first mode is called the enroll mode or training mode, and is used to set up a table of words (each is actually a bunch of digitized waveforms), and their meanings. This meaning may be a command to be activated or a textual translation of the verbal utterance. In this mode, some systems will average the data from several repetitions of each word before storing it in the table. In speaker dependent systems there is a separate table for each user, and a speaker who has stored voice patterns for words is "enrolled," while the system has been "trained" to recognize the voice of this user. The second mode is called active mode, and is used when the speaker wants the system to listen to and react to commands or sentences as spoken. This mode uses pattern matching. When the system digitizes the words spoken, it stores them in a buffer for analysis. It then compares this input pattern to all of the stored patterns in the table associated with the current speaker. There is typically no exact match, but frequently, one table entry is a much better match than any others. This best fit is taken as the word that the system recognizes as being spoken by the user. In speaker independent systems, users do not need to enroll. These systems try to recognize a word in their vocabulary regardless of who says it, so there is a single table used by all speakers. To accomplish this, some manufacturers have taken extensive samples of large groups of people saying the vocabulary words and then tried to create a template that represents the way most people say the words. The template in this type of system is usually stored in ROM. Because research into how we understand what we hear is still quite new, most of these speaker independent systems have quite small vocabularies. As our understanding increases, we expect the effectiveness and accuracy of these voice systems to increase.

Some voice input systems provide speaker verification or speaker identification. In a typical one of these systems, the speaker first types in an identifier such as the name and password. This tells the system which vocabulary table to use. The system then selects one of the vocabulary words and asks the user to say it. When the user speaks, the computer compares the input pattern to the one stored in the table. A particular person will usually say each word in a reasonably similar manner each time, and it is difficult for another to accurately imitate. This procedure is called speaker verification, and while it is not foolproof, it provides a modicum of system protection from intrusion.

The existence of voice documents and voice mail within office systems implies a need to alter and update these voice documents as can be done to text documents by text editors. Thus we briefly discuss voice editing

systems which allow one to insert, delete, modify and move voice segments. If these changes are constrained to occur on word boundaries, then this is helpful in reducing the precision with which an editing change must be specified, and results in a natural sounding message when segments are repositioned. A voice editor can use text labels to create an index for the voice message, so that the user can jump to any indexed point in the message without hearing the total message. The message can also be scanned by jumping forward or backward or by modifying the playback rate. The playback rate can be increased by decreasing the silent intervals between words. Approximately 20% increase in the playback rate is achieved through decreasing the silent interval between words by one sixteenth of a second. This does not affect intelligibility and the sentence structure can still be determined. This rate increase mechanism and others provide a voice analogy to speed reading. A time line of the voice message shown on a display screen helps a user to visualize editing changes, and to determine the position in the message while scanning. This interface has been found to be usable by novice computer users.

9.2. Primitive Text and Graphics Manipulation

In this section we consider how the bits representing either text characters or graphic objects can be efficiently manipulated using a raster display screen and a special instruction called RasterOp. The RasterOp instruction, called BitBlt on some computers, is a primitive which describes a two dimensional move data instruction for manipulating rectangular areas on a bitmap raster display. Parameters include size of rectangle to be moved, source, destination, and clipping region. Transformations on the bits of the region to be moved can also be specified in some implementations. This instruction is sometimes implemented in hardware, firmware, or software, and it under lies much of the flexible graphic and character manipulation available on modern high performance workstations such as a large selection of fonts and smooth scrolling of text.

If we want to show a picture on our bitmap display screen, this can be done rapidly if the picture is located in main memory as a bitmap denoting bitwise each pixel to be displayed. We use one RasterOp instruction to move these bits, specifying the destination as the portion of main memory which is used as the display buffer. Similarly, if we want to move a paragraph of characters on the screen to another position on the screen, we compute exact size, source, and destination; then we perform one RasterOp instruction. Note that source and destination can be overlapping parts of the display buffer memory, so the above move could be a small move upward representing a scrolling step.

The clipping region parameter specifies a destination area that limits the region affected by the RasterOp instruction, independent of the other parameters. Thus a large picture being partially viewed within a rectangular window will not disturb any region outside of its window if the window is used as the clipping region for all of the painting instructions and all scrolling instructions for this picture. By including this facility in the low level RasterOp primitive, clipping can be done rather efficiently, and it does not need to be replicated in all application programs.

Transformations which are possible in some RasterOp instructions include the capability to mix the bits from the source with the bits which previously resided at the destination. For monochrome (e.g. black and white) display systems, one can apply any of the boolean functions to combine the old bit for a particular pixel with the new bit for that pixel. Common functions are the replacement function, and the OR function. If D = old destination bits, D' = new destination, and S = source bits, then the OR function is defined as D' = SvD, and replacement is defined as D' = S. In some graphic illustration systems, the user has a mouse or stylus used for painting freehand on the screen. Replacement is called "painting over" and the OR function is called "painting under."

Sometimes it is useful to move the source figure to the destination while making that source appear some shade of grey rather than solid black. On bitmapped monochrome displays, this can be accomplished by using some regular pattern of alternating black pixels and white pixels (this works particularly well if the display is high resolution.) To this end, RasterOp sometimes incorporates a halftone parameter which is a small rectangle of bits which is uniformly repeated over the entire distination. This halftone rectangle can be combined with the source using any of the boolean functions. One frequent mode of usage is to AND together the source and halftone, which results in a source bit pattern masked by the halftone pattern. Also useful is a RasterOp instruction with halftone only (with no source) for painting grey backgrounds.

There are many interesting, useful, and sometimes complex macro operations which can be carried out via a simple sequence of RasterOp operations. For example, text can be made to look bold by ORing over itself shifted right by one pixel. We conclude this section by illustrating how the RasterOp instruction can be used for the rotation of arbitrary images by a multiple of 90 degrees. We can use RasterOp to rotate the four quadrants of the initial image; now it is only necessary to rotate each of the four smaller images. If we recursively apply this divide and conquer strategy, we obtain more and more regions of smaller and smaller size. Eventually, regions become single pixel size, and they are equivalent to their rotated images, so the recursion terminates with a completely rotated image. The steps of this algorithm would appear to take successively more computation steps, with the last one requiring several times more than (height x width) operations. However it is possible to perform all permutations for one stage in parallel so that the entire rotation can be performed with number of operations on the order of logarithm of (height) base two, i.e. $O(\log_2 h)$. The parallel permutation of many cells is accomplished with the aid of two auxiliary storage regions: the first carries a mask that selects the upper left quadrant of every cell; the second is used for temporary storage. A sequence of RasterOp operations exchanges the right and left halves of every cell, and then another sequence exchanges the diagonal quadrants, achieving the desired permutation. Many other image processing tasks can be performed with RasterOp. Dan Ingalls has built a complete optical character recognition system for sanscrit text using the various combination rules and operations that count the number of black bits in any rectangle. Bitmap processing is also well suited for VLSI implementation. The text and image operations expedited by RasterOp are so important in office information systems because these tend to be heavily graphic systems. It is valuable to be conscious of the RasterOp implementation, and any improvement in its operation that may be possible. For example, moving its implementation from software to hardware or expediting the execution of the simpler, frequently used versions of RasterOp, all tend to have handsome payoffs in system performance.

10. OFFICE ENVIRONMENT TOOLS

This section presents some of the *tools* useful to the designer/programmer in his/her construction of higher level office applications and systems. Many of these tools are general ones, useful to designers for building many kinds of software and applications, not simply office information systems. Since many of these topics are well-known to computer people (such as programming languages, and operating systems) we touch upon these topics in a general and brief manner, indicating those aspects which have been especially useful in design and programming of office information systems. Other tools (window packages) are described in more detail.

Many tools for software development are now generally available (for a price.) Since books have been written on tools for building general software [Kernighan], we will emphasize tools that are particularly useful for constructing office information systems. A viable strategy has been to provide an *open system* where separable tools can be added or subtracted by whomever has the need and implementation ability. Recall that this open systems property was one of the characteristics of offices emphasized in chapter 1. Generality is another characteristic which is desirable if the efficiency price is not too high. A very special purpose piece of code buried within and intertwined with a large piece of software will not likely gain widespread use. A little thought and careful modularization may go a long way toward usability. The UNIX system has emphasized the separability and generality of its utilities and has gained widespread popularity partly due to these characteristics; we will discuss UNIX and other systems in this chapter. Separability and generality will be themes emphasized in this chapter and throughout the text.

Primary basic tools are programming languages and operating

systems. We will overview the notion of generations of these tools, comment on some recent advances useful to office information system construction, and discuss programming environments and window packages. Later we will apply the notion of generations of systems to office information systems.

10.1. Programming Languages and Compilers

Programming languages are the primary tool used by programmers to implement office information systems. We note that all programming languages reside within a spectrum between low level languages and high level languages; and between general purpose languages and special purpose languages. There are very low level languages such as microcode instructions and machine language, and a spectrum of higher level languages. Languages like machine and assembly languages describe the computer's behavior in a very machine dependent manner, and so are called machine dependent or low level languages. These were the only type of language available on the first stored program computers, and thus are also called first generation languages.

Fewer applications are written today in assembly language because of higher level languages such as Algol that can match the efficiency of assembly language with less programmer time needed. Optimizing compilers for high level languages can sometimes generate better machine language code in less time than the average programmer writing in assembly language Thus a large amount of application code is written in COBOL for business data processing applications and in FORTRAN for scientific applications. These high level languages are sometimes called second generation languages, and although there is not universal agreement, there are definitions in the literature of third, fourth, and fifth generation languages. We will comment on the utility and relevance of each of these generations for office information system design.

Many office information systems have been built in third generation languages such as Pascal or C which provide more modern control structures and data structuring than second generation languages. This third generation of structured programming and object oriented programming, developed during the 1970s, has introduced numerous advances such as enhanced portability, data abstraction, and class/inheritance mechanisms. When investing in the implementation of any large software system (some office systems are quite large), it is useful to carefully consider one's portability goals. If written in a high level language, there is potential that since it is machine independent (relatively), it can be executed on many different computers. However, some languages assume a particular type of architecture and may run inefficiently on a radically different machine - many use a stack machine model which can often be implemented efficiently; most assume a Von Neumann machine of some kind. Even purportedly compatible versions of a language running on different machines have been notoriously troublesome. Particularly since earlier languages tended to be incompletely specified so different compilers performed different actions in these ill-specified cases.

Programming languages with data abstraction allow the programmer to define her/his own data types using the base data types and previously defined data types. These languages also allow the association of a set of operations with each data type. A program module may then choose to allow access to the data by other modules only through use of the defined operations.

Programming languages with classes and inheritance expand upon the notion of abstract data types by allowing hierarchies of types of abstract data plus code - called objects. An object has structure and properties determined by the class or classes to which it belongs. This is similar to the notion that each datum has a data type, however, since objects are grouped into a hierarchy of classes, an object can "inherit" much of its structure and properties from its classes. Note that if we want to override some particular class property or behavior for some of its objects, we simply introduce a subclass of the class for this purpose. We need specify within the subclass only those properties which we want to be different from the original class; all other properties are automatically inherited by all members of the subclass. Object oriented systems with multiple inheritance have also been implemented.

The fourth generation has been called the generation of applications generators. These languages have been developed primarily during the late 1970s and 1980s in recognition of the "software crisis" in which there is a much larger need for application specific systems than can possibly be implemented by the limited number of programmers in the world. Thus, some pieces of software have been specifically designed to encourage the unsophisticated computer user to quickly develop her own data-intensive programs, and thereby significantly improve software productivity. Fourth generation applications generator languages include Nomad, Focus, the visicalc-like systems, the DBase-like systems, and QBE/OBE (the office by example system.) Note that all of these languages assist the end user in quickly building application systems without professional programmers and without using COBOL, FORTRAN or ADA. This domain is properly the topic of our next chapter where we discuss application facilities for use by the end user.

The fifth generation of languages is characterized by very high level general purpose languages. Many of these are nonprocedural languages or functional languages. Most of these languages such as LISP have automatic garbage collection. Some of these languages such as Prolog have built in inferencing engines for knowledge based processing. Many of these languages such as Smalltalk present a programming environment rather than simply a language. Thus knowledge of the language is equivalent to knowledge of the debugger, of the editor, of the operating system commands, and of the database language because they all present the same interface and are integrated together. All of these languages have a multiplicity of advanced features, and allow rapid prototyping. Their efficiency has been steadily improving so that they are now considered serious development systems.

10.2. Operating Systems

Operating systems are the fundamental control programs and structures that underlie many of the tools and applications built by programmers. The operating system controls the computer hardware, manages system resources, runs programs in response to user commands, and supervises the interaction between the system and its users. The operating system forms a foundation on which applications software, such as word processing, spreadsheet, and acccounting programs, is developed and executed.

On the earliest stored program computers, there were no operating systems. Typically scientists signed up for a slot of time, and manually loaded their own program into the computer, loading registers, and generally performing all the tasks of job sequencing that later were performed by operating systems. Indeed the first generation of operating systems were primarily job sequencers for batch processing. These sufficed to get the programmer out of the machine room and to efficiently run one job after another.

From about 1959 to 1963, several hardware advances came into use and stimulated a second generation of operating systems. One of the most significant innovations was the "data channel" - a primitive computer that controls the communication and data transmission between the main computer and IO devices. This together with the "interrupt" stimulated the construction of complicated input output programming systems written to take advantage of the new architectural features. These included software buffering routines to permit automatic reading ahead of jobs and queuing output for delayed printing. Interrupt routines were written to respond to IO interrupts and to then return control to the interrupted program. However, one job at a time processing still resulted in low channel utilization for compute bound jobs, and low CPU utilization for heavy IO jobs.

From about 1962 to 1969, a new method of running jobs, multiprogramming, came into almost universal use in large operating systems, and a new way of computing, timesharing, was also developed as an alternative to batch processing. Some of these third generation systems grew into very large operating systems; in some cases it was found that the overhead introduced by the operating system outweighed its advantages. The fourth generation was typified by CP/M. It was the opposite extreme of the large third generation operating systems. Stimulated by the spectacular rise to popularity of the personal computer, there was again demand for single user, small operating systems. CP/M (Control Program for Microprocessors) was originally developed in the 1970s for the Intel 8080. It became the first defacto microcomputer standard when other microcomputer

manufacturers also seized CP/M as the solution to their software problems. CP/M offered a good compromise operating system for the microcomputers of that period because of the following features:

- It was small, requiring only 8K of main memory.
- It handled low level input output tasks,

freeing programmers to concentrate on their applications.

- It was portable with hardware specific functions concentrated in

one small part of the software.

- It was relatively simple and easy to learn.

There are many systems which have been used as the base for interesting useful integrated office information systems, and we mention two, Pilot and UNIX, which strike a happy medium between the tiny and the gigantic.

10.2.1. The Pilot Operating System

Pilot is an operating system designed for the personal computing environment, and used as a base for the Xerox Star office information system introduced in 1981. It provides a basic set of services within which higher level programs can more easily serve the user and communicate with other programs on other machines. Note that Pilot does not provide direct facilities to the user such as a command language. Similarly other functions commonly found in modern operating systems such as character string naming of files are absent or relegated to higher levels. On the other hand, Pilot provides a more complete set of services than is normally associated with the "kernel" or "nucleus" of an operating system. Much of the design of Pilot stems from an initial set of assumptions and goals rather different from those underlying most time sharing systems. Pilot is a single user, single language system, so it has only limited features for protection and resource allocation. Pilot's protection mechanisms are defensive rather than absolute since in a single user system, errors are a more serious problem than maliciousness.

Pilot supports high bandwidth, close user - system cooperation. It assumes a resource rich personal environment including a high resolution bit map display, a 2**32 word virtual memory, a flat file system, and streams for serial input and output. Pilot is closely coupled with Mesa, a high level systems programming language. It is entirely written in Mesa, and Mesa depends upon Pilot for much of its run time support such as its parallel processes and monitor facilities. Since Mesa is the only language supported, many of the language independence arguments that tend to maintain distance between the operating system and the language are not relevant. Interaction between loosely coupled process on different machines is supported by the Pilot communications facility. This facility allows processes on different machines to communicate with each other via a hierarchically structured family of packet communication protocols. Communication software is an integral part of Pilot, rather than an optional addition, because Pilot was designed to be a suitable base for network based distributed systems.

10.2.2. The Unix Operating System

The Unix family of operating systems, developed at Bell Telephone Labs in the 1970s, spans a range of sizes that fit on microcomputers, minicomputers, and mainframe maxicomputers. UNIX is well suited for software development, and is intermediate in complexity between the very small operating systems such as CP/M and the large operating systems such as Multics. Although originally designed as a single user environment, UNIX is now available as a multiuser system. It thus supports efficient sharing of the processor and of the information storage of a computer system, while offering the security and protection features needed to insulate each user from the acitvities of other users. UNIX based application systems with their ability to support many users working with common data at relatively low cost per user can be a good match for use in smaller offices and departments of large companies.

Due to its high portability, UNIX is available on hundreds of different computers. Moving UNIX to a new system typically requires only a few man-months of effort. This vendor independence of the operating system and of applications software built on top of it, is one of the major factors influencing the UNIX proliferation. Another major factor is the large set of utilities available with UNIX; over two hundred utility programs for functions like sorting data, processing text, and searching information allow many tasks to be performed by concatenation of these functions without writing new programs. The UNIX pipe is a facility that can be useful in combining programs or tools to perform more complex functions. UNIX is particularly rich in text processing tools and software development tools.

UNIX supports background processing which allows a user to initiate a task, and then proceed to other activities, while the system continues to work on the original task. For example, the system can be sorting a file and printing a report on a user's behalf at the same time that the user is editing a document. As opposed to Pilot, UNIX features a hierarchical file system. All of these features are accessible to the user by use of the UNIX shell. The shell is a powerful command interpreter that provides a number of features such as the ability to redirect application input and output, and the ability to manipulate groups of files with a single command. The shell also supports execution of predefined command sequences in conjunction with built-in programming language features. The major complaint of office users is that the UNIX shell is "cryptic and unfriendly." UNIX is a solid, time tested operating system, which has been in use for over fourteen years. The software is mature and relatively free of bugs, offering a high level of reliability for an operating system of its capability. On the other hand, UNIX is a third or fourth generation system, but not what I would think of as a fifth generation operating system. A fifth generation operating system needs to be a distributed network operating system; perhaps knowledge based, and definitely friendly and unobtrusive. It should be a coherent part of the office work environment, not identifiable as a separate entity and separate set of commands. This leads to the notion of a *programming environment* in which the programming language, the operating system, and other elements live in harmony, as discussed in the next section.

10.3 Programming Environments

Some of the fifth generation languages, as mentioned in section 4.1, have blossomed into programming environments. These environments provide a convenient uniform interface to a host of tools, sometimes called a programmer's workbench, such that the language constructs and mode of operation is the same whether one is using functions of the compiler, the operating system, the debugger, the database, or the editor. The system will frequently remember the state of various pieces of code, provide tools for configuration analysis and generation, and handle a lot of the bookkeeping associated with the maintenance and updating of large (thousands of modules) software systems such as office information systems. In these environments, we see that there is frequently no explicit entity called the operating system, although these functions still get performed explicitly, or more frequently, implicitly behind the scenes.

Lets follow Anna Ying, a programmer, as she does some design, some coding, and some debugging within her fifth generation programming environment. A number of these systems make use of one or more color bit map displays with multiple *windows* (see next section) to allow the programmer to see multiple files, or optionally to work on multiple tasks. Some of these systems also allow the programmer to specify commands by pointing to icons or by speech input. Within a typical session, Anna may first browse through the database of current program modules, and see that there are three modules that she has been working upon recently which are in various states of incompleteness. Any programmer with proper project access authority may select one of these modules to work on, and the environment will check this module out of the *program library* if it is designated as a non-shared module, or generate an experimental version for updating while other programmers will still have read / execute access to the previous (already tested) version. A number of tools for version control, consistency checking, and compilation of libraries of modules are available whenever needed. The system immediately shows Anna a structural graph of the module she is working on together with its properties and relations to other modules. Anna opens another window to see a list of the most recent changes to this module; she particularly takes note of the alterations that have been made by other people to the module since she last worked upon it, and asks the system to italicize those altered parts of the code whenever she looks at them. After displaying a high level view of her code, Anna zooms in on one particular line of code that she wants to change. She asks tha system to

show her where the variables present in this line are used, and after examining these in other windows, she makes the change. Since the editor is a syntax directed editor, the change she makes is guaranteed to be syntactically correct. Anna then constructs a little test environment and tests the code that she just altered. During this construction, Anna can view many different critical pieces of code, data, variables, and structures by opening many different windows. During the test run, the system pauses and verbally asks for test data values. After providing these values, Anna opens another window to work on a separate document editing task that requires copying a table of information from a previous electronic mail message into a bar chart in the document; she annotates this with a short animation, distributes this to the programmers in her group, and makes a note in her electronic calendar that this document must be again updated after end-of-the-year data arrives. The window which was testing the altered module beeps and displays a short results message when it finishes. Anna carefully peruses the details of the results, requesting to see snapshots of intermediate program states, and invokes the debugger to trace down the source of one incorrect case value. After she is satisfied with the program results, she sends a message to Robert to please test this module with his extensive test environment. She instructs the environment to bind this new version to Robert's environment for testing, and notify her after Robert has completed his tests. If the tests are successful, then the system should automatically recompile this and any other modules which must change as a result of Anna's changes to form a new release system. Note that within this scenario, many general office information tools and system elements were utilized which are not specifically programming tools. The mail system, the database system, the notify system, and the calendar are all examples showing that many specialists such as programmers can make good use of many generic office information system features.

10.4. Window Managers

In this section, we will discuss the idea and architecture of window management systems. These are subsystems used both as tools by programmers, such as the example above, and also used as task execution facilities by end users who are not computer experts. A *window*, as present within many display based office information systems, is a region of the display screen which has a well defined border, and is used for interaction with the user on one specific task. Windows are significant because they present the user with a natural model of parallel, asynchronous activity. Prior to the advent of windows, there was no simple, graphical, easily understood method for the casual user to specify that he or she wanted to work on two tasks or two documents at the same time, and track their progress. With modern window systems, I can be reading and retrieving my electronic mail at the same time that I am doing some financial calculations with my spreadsheet, because each of these can be visible on the display screen at the same time within separate rectangular regions called windows. Of course, one could say that windows were always available on display systems, because the whole screen can act as a single window. To avoid this degenerate case, we qualify a window management system as one which supports two or more simultaneous windows; and indeed, in some systems, these window regions are not restricted to being rectangular. Once we specify that windowing means support of more than one simultaneous active region on the screen, we have the need for coordinating which events should be reflected in which window. One approach to satisfying this need is by means of a system program called a *window manager or windowing package*. This is a particularly attractive solution if the number of simultaneous windows can be large, and the system presents flexible means of moving and manipulating windows. Operations that a flexible windowing package may support include the following.

- Create a window
- Destroy a window
- Move a window
- Shrink or enlarge the size of a window
- Maximize (full screen) or minimize (iconic) the size of a window
- Reshape a window
- Place a window on top of all overlapping windows
- Place a window beneath all overlapping windows
- Find or activate or deactivate a window

Many other operations are possible on windows, so it is useful to have a window manager to coordinate these many functions. For example, there are a set of actions that may be performed upon the content of a window. These actions include displaying and associating an object with a window, erasing and dis-associating an object from a window, horizontal and vertical scrolling of the object displayed in the window, editing and changing the view of an object within a window, and manipulating sub-windows and linked window contents.

The user interface presented by the window system is also an important aspect. Some window systems have permanent fixed size and position for each window on the screen, and some do not. Some window systems allow overlapping windows, and some do not. Some window systems invoke the above operations by keys on the keyboard, a mouse button, or other devices. Some systems have a header at the top of each window with the label of the window and all commands. Others choose to put a command line at the bottom or side of each window which may show only the subset of commands which can be invoked given the current state. Others use pop-up menus to invoke operations. Each of these have certain tradeoffs of simplicity, convenience, and speed of usage.

The generic window management architecture which we present here is a software/hardware system consisting of three modules: the inter-window handler, the intra-window handler, and the virtual screen handler. These modules interact with each other and with higher level systems such as application handlers. They also interact with lower level systems such as input handlers and output drivers within the operating system. This interaction is illustrated in figure 4. Inside the box labelled *window handler* which represents the window manager, are three more boxes representing the three modules mentioned above. Only the modules inside the window handler box need to know about the window data structures which specify the visibility characteristics of each object with respect to its windows, and the visibility characteristics of each window with respect to the screen and with respect to the other windows.

Thus application processes, which are shown outside of the window manager in figure 4, can deal with events related to their viewed objects as if they owned the whole screen, and as if their windows showed the entire view of the complete object (called virtual views). They need not worry about partial views due to window size and shape, and clipping due to other windows partially or totally obscuring their windows. Coordinate (x_r,y_r) points are passed to and from the applications processes in relative coordinates - that is, coordinates of the virtual view. Since the application processes operate with these more abstract coordinates and views, we call these processes higher level systems.

Also shown in figure 4 are boxes labelled input handler and output driver which somehow connect either directly or indirectly to actual hardware devices for input and output such as keyboard, mouse, and display screen. The function of the input module is to field interrupts from one or more devices and queue and encode them to be passed along to the window manager. This allows the window manager to receive them in a more civilized manner such as a queue of messages, rather than being interrupted in the middle of other important tasks. The manager must simply insure that the messages are taken out of the queue and processed within a reasonable amount of time. Similarly, the output driver is receiving messages and turning these into appropriate signals to drive devices or to interact with channels or output processors. Although we have stated that operating systems may not be explicitly separable in modern systems, the functions performed by these boxes (I/O) are typical operating systems functions, so that the window manager can be considered to interface with an operating system or some similar lower level system. We see that the window package is positioned between the operating system level and the application level to allow multiplexing of interactions by the user with various application processes. These input and output modules communicate with the window manager in terms of absolute screen (x_a,y_a) coordinates, and do not need to worry about in which window a screen event, such as a mouse button click, occurs. These modules need not be concerned with the meaning of events, nor which application process is related to an event. Classes of events recognized by the window manager may include position events (e.g. mouse or eye movement), character events (e.g. keyboard), and choice events (e.g. menu selection or voice command).

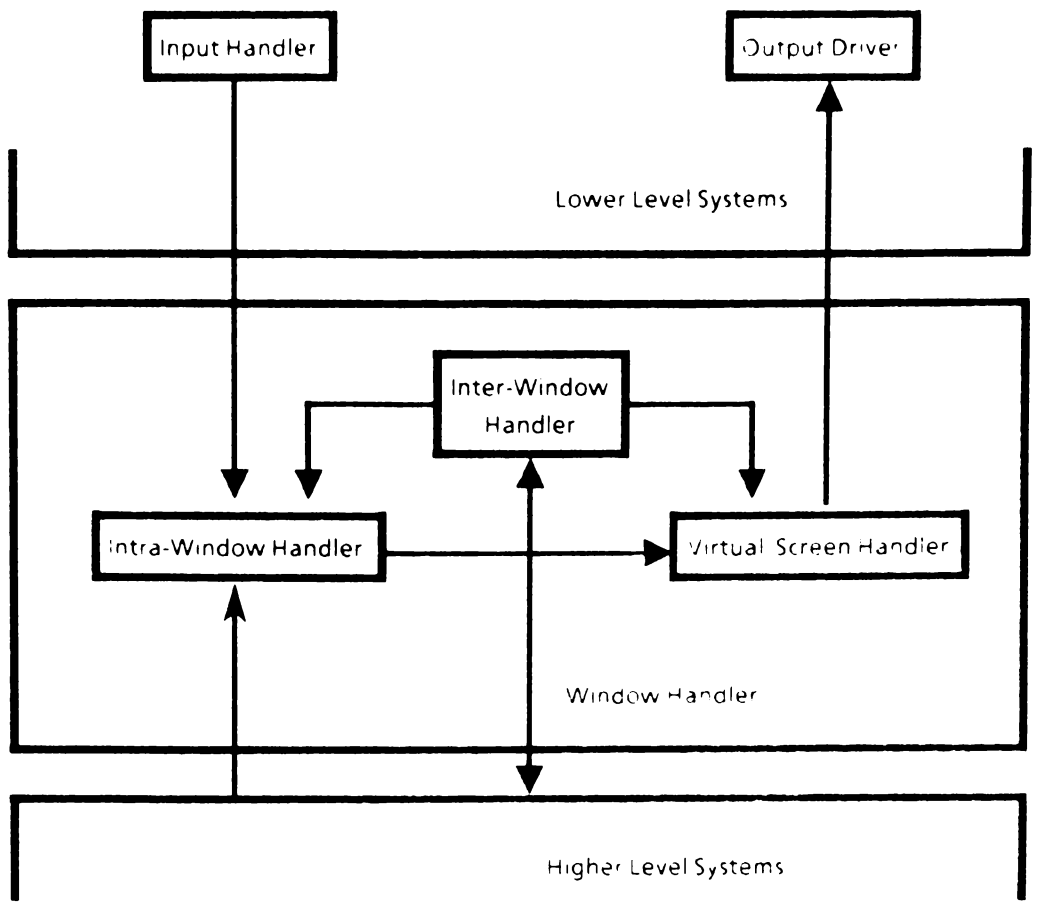


FIGURE 4

Window Management

An event such as the pressing of a mouse button might imply the selection of a character within a window. An event such as this is passed into the window manager by first going to the inter window handler which must decide which window or group of windows are interested or affected by this event. There is a translation of the cursor coordinates which accompany the mouse click from absolute coordinates to relative coordinates by this handler before it passes the information along to the correct application processes representing the relevant windows. Figure 4 shows this by an arc going into the inter window handler from a lower level process, and an arc leading out to a higher level process. Note that menu processes are also higher level processes, and can be treated identically to application processes. The receiving application process determines the meaning of the event. If this mouse click means to move a window, then the application process again invokes the inter window handler with positioning information. Alternatively, some inter window handlers handle generic windowing operations such as this without notifying or disturbing the application process. In any case, the inter window handler manipulates data structures indicating an adjustment in the relative positioning of all affected windows. Next the screen handler is invoked. Whenever this module is called by either the inter or intra window handler, it evaluates the current data structures of what windows are partially or totally on top of what other windows and directs output drivers to make screen updates occur. Note that windowing can be implemented on most any type of display, but some will be slower and clumsier than others. For example, if it is necessary to rewrite the whole display to update a small region, this is considerable overhead.

Suppose on the other hand that this mouse click meant "select a character inside a window for editing," then the application process would invoke the intra window handler, because this operation involves updating the contents of one particular window. The application may decide to perform video reversal, or to continuously blink the character. This message can be transmitted via a rather high level language to the intra window handler which must decode the message, translate from relative to absolute coordinates, and perform the action. Next the screen handler is invoked to update the screen to show the feedback of video reversal or blinking. This would complete the sequence of invokations stimulated by that single mouse click.

A good window package suppports direct and indirect interaction between different windows. For example, if two windows are overlapping views of data within a database, then update to the first of these windows may warrant update of data visible within the second window. This second view may be quite different from the first. It may, for example, show a bar graph of information which is in table form in the first window. This implies the dynamic and automatic adjustment of the height of one or more bars. In general this may be a computationally heavy view transformation, and may consume a lot of time. Therefore, this update to the other window can occur immediately (may in some cases slow things down immensely), or upon request by the person watching the screen, or as a background task performed when there is idle time. Thus, figure 4 shows an arc from the inter to the intra window handler; this link is also useful for handling errors and exceptions.

An option within this design that has been implemented within research environments is the hierarchy of window handlers. This means that the higher and lower level systems with which a window handler interacts may also be window handlers. Thus, a window allows one to view part of one or more objects (called a *scene*) through a *viewport* at a lower level, which may in turn be windows displaying views of various scenes at even lower levels. Thus one can view transformed parts of many different scenes within a single window. It is then useful to be able to move, expand, contract, and rotate either a window, or the scene which is being viewed through the window. This facility is quite useful within CAD/CAM systems, within geographical display systems, as well as within office information systems.

In concluding our discussion of window management systems, we state that the window handler is a special case of the multiprocess interactive input/output handler. It relies heavily for its effectiveness upon bit map displays and RasterOp instructions. One could envision similar primitives being discussed to support other media such as voice

processing and interfacing.

The design presented is general and modular. Only modules in the window handler know about and access the window data structure; only code within the inter window handler actually changes the window structure; only the screen manager knows the details of the screen bits. These and other considerations make the design modular. Applications can be constructed as if they owned all of a large screen with no concern for the presence of other windows or absolute coordinates. Furthermore, the modularization allows the screen module to be simulated easily, and display hardware considerations and changes to be rather isolated. This all encourages the good practices of easily keeping logs of activities at the higher abstract operation level, and of separating operations within a window from bookkeeping operations between windows. Thus one can implement recovery, replay, and undo schemes.

The generality of this design allows multiple screens, and multiple input/output devices to coexist. If the window handler is written as reentrant code so that many calls to it can be simultaneously outstanding, then there can be many variations and replications of the intra window handler and of the screen handler. However, implementations usually allow only one copy of the inter window handler to guarantee integrity of the window data structures. Menus are handled in a general way *outside* of the window manager, so there can be multiple menu handlers, and these can even support several user interfaces simultaneously (although I don't recommend it.) Some systems arrange there menu handler as a separate layer between the window manager and the application processes. This fits in well with our window handler design. Some systems arrange windows (which are associated with processes) in a tree structure with a maximal background window covering the whole screen as the root node. This also fits in well with our window handler design. Many commercial window systems have lots of other parameters associated with windows, and many implement subwindows. One question which occurs in a system as general as ours is: How do you know which window receives a keystroke or event when there are multiple focal points, multiple screens, multiple selection points, and multiple type-in points potentially active simultaneously? Typically this is determined by some mapping from input/output devices to windows, sometimes called pipe connections, and which can be dynamically changed. Some systems use defaults stored within a user profile, and others use a set of prioritized tables. This latter choice can even be tailored so that different key strokes from a single input device can go to different windows as a function of the key combination and the "state" of the window.

Generality sometimes extracts a cost in terms of implementation complexity and speed of execution. Complexity can actually be decreased by modularity such as exhibited by our design, but some systems have been forced to short cut some of the parts of the design. For example, an appplication process may directly access the screen manager for efficiency in certain cases. It is clear that the speed and power of future generations of microprocessors will not be wasted with idle cycles in the domain of office information systems.

11. SUMMARY AND CONCLUSIONS

We have tried to compose an introduction to office systems which presents some information and intuition about the office, its people, and its technology. Our emphasis has been on the human element and its interaction with technology. In keeping with this emphasis, we introduced a general framework which makes explicit the layered processes and protocols which occur between individuals and their personal environments; also between groups, organizations, and their communal environments. A framework such as this can be applied to a wide variety of problems [52] such as (1) explaining historical changes in the structure of American business organizations, (2) predicting changes in the structure of human organizations that may result from the widespread use of computers, and (3) analyzing the advantages and disadvantages of decentralized task scheduling in computer networks. It is noteworthy that the interdisciplinary nature of this field encourages techniques of one area (e.g. organizational design to be applied to problems in seemingly unrelated areas (e.g.computer science). We conclude this chapter with a few statements about integrated systems, current and future.

The ability to easily pass information between applications, and the ability to use the same interface, data structures, and commands for various applications has greatly increased the utility and ease of use of office information systems. One example is the table of data which can be displayed in a document, and also used to generate a bar chart and a pie chart and other data items in the same or another document. When an entry in the table is changed, the bar chart, the pie chart, and the other data items are all automatically updated correspondingly. Finally, we note that a number of conversion services are becoming available to convert documents from one format to another so that documents can be viewed and edited within heterogeneous environments. Also, gateway services are becoming available. These are computer systems which are connected to more than one network; they handle the translation and buffering needed in order to allow information to flow between heterogeneous networks. This progress with converters and integrators sets the scene for the next generation of knowledge based office assistants which will be able to combine local user preference, global office knowledge, and inference to assist with many crucial functions. These computerized personal assistants knowing organizational and social structures and their relations to information and procedures, can make sure that global office procedures are handled by the numerous parties involved in a timely fashion, and that crucial meetings are not forgotten. It is becoming increasingly apparent that office models which take into account *people in offices* and *technology in offices* are needed for the successful design and installation of office information systems.

12. REFERENCES

1. ACM Computing Surveys. Special Issue: Psychology and Computer Systems. September, 1980.

2. Aderet, A. Menu-Driven and Command-Driven Word Processor Interfaces: Evaluation of Ease of Learning. Proceedings of the 1982 Office Automation Conference. 1982 April; AFIPS Press. 875-884

3. Allen, J. Synthesis of Speech from Unrestricted Text, IEEE Transactions. 1978 April; 64 (4): 433-442.

4. Allen, R.B. Composition and Editing of Speech. Bell Systems Technical Journal, 1981.

5. Allen, T.; Nix, R.; Perlis, A. PEN: A Hierarchical Document Editor. ACM SIG-PLAN SIGOA Symposium Text Manipulation. ACM SIGPLAN Notices. 1981 June: 16 (6): 74-81.

6. Alter, S. Decision Support Systems: Current Practice and Continuing Challenges. Reading, Massachusetts: Addison-Wesley; 1980.

7. Argyris, C. Management Information Systems: The Challenge to Rationality and Emotionality. Management Science 1971; 17 (6): 275-292.

8. Bair, J.H. Communication in the Office of the Future: Where the Real Payoff May Be. Proceedings of the International Computer Communications Conference: 1978 August; Kyoto, Japan.

9. Bair, J.H. Productivity Assessment of Office Automation Systems (two volumes). Report for the National Archives and Records Service, SRI project 7823. 1979 March; Menlo Park, California.

10. Bair; Farber; Uhlig. The Office of the Future. North-Holland. 1979:379.

11. Barber, G. Reasoning About Change in Knowledgable Office Systems. Workshop on Research in Office Semantics, Chatham, MA, 1980 June.

12. Bolt, R.A. Put-That-There: Voice and Gesture at the Graphics Interface. SIGGRAPH '80 Conference Proceedings, 1980 July: 262-270.

13. Booz; Allen; Hamilton. The Booz Hamilton Multiclient Study on Office Productivity. New York, 1980.

14. Bracker, L.; Konsynski, B.R. A Model of the Automated Office. MIS Technical Report, Department of Management Information Systems, University of Arizona, 1980.

15. Business Magazine. Special issue on III-Structured Problem Solving. Jan-Feb 1979.

16. Card, S.; et al. The Keystroke Level Model for User Performance Times with Interactive Systems CACM, Vol. 23, No. 7, 1980.

17. Card, S.; et al. The Psychology of Human-Computer Interaction. Lawrence Erlbaum Associates, 1983.

18.Chang, J.M.; Chang, S.K., "Database Alerting Techniques for Office Activities Management", IEEE Transactions on Communications, January 1982, 30(1); 74-81.

19. Cohen, D. A Protocol for Packet-Switching Voice Communication. Computer Networks 1978 September/October; 2 (4/5): 320-331.

20. Computer Message Systems. Uhlig. ed. North-Holland. IFIP , 1981.

21. Conrath, D.W. Communications Environment and Its Relationship to Organizational Structure. Management Science. 1973 December 20 (4): 586-603.

22. Conrath, D.W., and Bair, J.H. The Computer as an Interpersonal Communication Device: A Study of Augmentation Technology and Its Apparent Impact on Organizational Communication. Proceedings of the 2nd International Conference on Computer Communications. Stockholm, Sweden, August 1974.

23. Cortese, G.; Sirovich, F. A Daemon-Based Programming System for Office Procedures. Proceedings of the Second ACM-SIGOA Conference on Office Information Systems; 1984, June 25-27: Toronto, New York: ACM; 1984; 2: 203.

24. Cotton, I.W. Technologies for Local Area Computer Networks. Proceedings of the LACN Symposium, May 1979, 24-45.

25. Coutaz, J. Towards Friendly Systems, Design Concepts for Interactice Interfaces. 1984 April; Proceedings of the 1984 Southeast Regional ACM Conference, Atlanta 56-61.

26. Czedo, B.; Embley, D. Office Form Definition and Processing Using a Relational Data Model. Proceedings of the Second ACM-SIGOA Conference on Office Information Systems; 1984, June 25-27: Toronto, New York: ACM; 1984; 2: 123.

27. De Jong, S.P. The System for Business Automation (SBA): A Unified Application/Development System. Proceedings of IFIP 80, October 1980.

28. Ellis, C.A. Information Control Nets: A Mathematical Model of Office Information Flow. Proceedings of the 1979 ACM Conference on Simulation, Simulation, Modeling and Measurement of Computer Systems. August 1979, 225-240.

29. Ellis, C.A.; Nutt, G.J. Computer Science and Office Information Systems. Computing Surveys, 1980 March; 12 (1): 26-60. (Also available as Technical Report, Xerox Palo Alto Research Center, Palo Alto, CA. June 1979.)

30. Embley, D.W. A Natural Forms Query Language-An Introduction to Basic Retrieval and Update Operations. Scheuermann, P. ed. in Improving Database Usability and Responsiveness, Academic Press, Inc. 1982. 121-145.

31. Engelbart, D.C. Augmenting Human Intellect: A Conceptual Framework. Summary Report, SRI, Menlo Park, California, 1962.

32. Engelbart, D.C.; English, W.K. A Research Center for Augmenting Human Intellect. Proceedings of the Fall Joint Computing Conference December 1968. AFIPS Press. 395-410.

33. Gibbs, S. Office Information Models and the Representation of Office Objects. Proceedings of the ACM Conference on Office Information Systems, 1982.

34. Good, M. Etude and Folklore of User Interface Design. Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation, Portland, Oregon, June 1981.

35. Greif, Irene. The User Interface of a Personal Calendar Program. Proceedings of the NYU Symposium on User Interfaces, New York University, May 1982.

36. Gunther, K.D. PLOP--A Predicative Programming Language for Office Procedure Automation. IEEE Workshop on Languages for Automation, Chicago, November 1983.

37. Hammer, M.; Zisman, M.D. Design and Implementation of Office Information Systems. Office Automation, Infotech State of the Art Report. Series 8, No. 3. Infotech Limited, Maidenhead, Berkshire, England 1980.(Also in Proceedings of the NYU Symposium on Automated Office Systems, New York University Graduate School of Business Administration. May 1979, 13-24.)

38. Harris, L. and Associates, Inc. The Steelcase National Study of Office Environments: Do They Work? 1978. Available in Canada from Steelcase Canada Ltd. P.O. Box 9, Don Mills, Ontario M3C 2R7.

39. Herot, C.F. Spatial Management of Data. ACM Transactions on Database Systems. 1980 December; 5 (4): 17-32.

40. Hiltz, R.S. Computer Conferencing: Assessing the Social Impact of a New Communications Medium. Technological Forecasting and Social Change, 1977 10, 225-238.

41. Hiltz, S.; Turoff, M. The Evolution of User Behavior in a Computerized Conferencing System. Commuunications of the ACM, 1981 November; 24 (11): 739-751

42. International Standards Organization ISO/TC97/SC16 Reference Model of Open System Interconnection ISO/DIS 7498, International Organization for Standardization, December 1980.

43. Israel, J.E.; et al. Authentication in Office System Internetworks. ACM Transactions on Office Information Systems. 1(3):193-210; 1983 July.

44. Jacob, R.J.K. Using Formal Specifications in the Design of a Human-Computer Interface. Human Factors in Computer Systems Proceedings (March 1982), 315, 321.

45. Kay, A.C. Microelectronics and the Personal Computer. Scientific American, 1977 September; 237 (3): 231-244.

46. Ke, J.S.; et al. SLOS: A Specification Language for Office Systems. IEEE Workshop on Languages for Automation, Chicago, November 1983.

47. Kease, W.J. Towards an In-House Integrated Publishing Service. Proceedings of the Second ACM-SIGOA Conference on Office Information Systems; 1984, June 25-27: Toronto, New York: ACM; 1984; 2: 113.

48. Keen, P.G.W.; Morton, Scott M.S. Decision Support Systems: An Organizational Perspective. Addison-Wesley, Reading, Mass., 1978.

49. Kenney, G.C.; et al. An Optical Disk Replaces 25 mg Tapes. IEEE Spectrum 16(2), 33-38, February 1979.

50. Le Lann, G. Control of Concurrent Accesses to Resources in Integrated Office Systems. Integrated Office Systems-Burotics, ed. Naffah, North-Holland, 1980.

51. Limb, J.O. Integration of Media for Office Services. Office Automation Conference Digest, 1981, Houston, TX March 1981, 353-355.

52. Malone, T.W; Smith, S.A. Tradeoffs in Designing Organizations: Implications for New Forms of Human Organizations and Computer Systems. MIT Center for Information Systems Research Report SLOAN WP #1541-84, March 1984.

53. Maslow, A.H. Motivation and Personality. NY: Harper, 1954.

54. Maxemchuk, N.F. An Experimental Speech Storage and Editing Facility. B.S.T.J. 1980 October; 59 (8): 1383-1395.

55. McLean, R.J. Organizational Structure as Viewed by Intra-Organizational Communication Patterns. Ph.D. Thesis, University of Waterloo Management Sciences Department, Waterloo, 1979.

56. McQuillan, J.M. Local Network Technology and the Lessons of History. Proceedings of the LACN Symposium, May 1979, 191-197.

57. Meisner, N.B. The Information Bus in the Automated Office. Integrated Office Systems-Burotics, ed. Naffah, North-Holland, 1980.

58. Melnyk, V. Man-Machine Interface: Frustration. Journal of the American Society of Information Sciences. 1972; 23 (6): 392-401.

59. Mintzberg, H. The Nature of Managerial Work. Harper & Row, New York. 1973.

60. Morgan, H.L. The Interconnected Future: Data Processing. Office Automation, Personal Computing. Proceedings of the LACN Symposium, May 1979. 291-300.

61. Morgan, H.L.; Buneman, O.P. Implementing Alerting Techniques in Database Systems. Proceedings of the AFIPS National Computer Conference, 1977.

62. Morgan, H.L.; Soden, J.U. Understanding MIS Failures. Proceedings of the Wharton Conference on Research on Computers in Organizations, Database, New York: ACM, Vol. 5:2,3,4, 1973, 157-171.

63. Newman, W. Office Models and Systems Design. Naffah, N. ed. in Integrated Office Systems-Burotics, North Holland, Amsterdam (1980) 3-10.

64. Ouchi, W. A Conceptual Framework for the Design of Organizational Control Mechanisms. Published in: Connor, P.,ed Organization Theory, A Design Perspective. SRA Inc, 1979.

65. Ouchi, W. Theory Z. Addison-Wesley Publishing Co., 1981.

66. Pilote, M. The Design of User Interfaces for Interactive Information Systems. PhD Thesis, Department of Computer Science, University of Toronto, 1982.

67. Porat, M.Y. The Information Economy: Definition and Measurement (Nine Volumes). U.S. Government Printing Office, Washington, Washington D.C., July 1977.

68. Propst, R. The Office: A Facility Based on Change. Zeeland, Michigan: Herman Miller, Inc., 1968.

69. Reichwald, R. Cooperation in the Office-Office Communication Systems as a Management Tool. Proceedings of the Second ACM-SIGOA Conference on Office Information Systems; 1984, June 25-27: Toronto, New York: ACM; 1984; 2:212.

70. Richer, I. Voice, Data and the Computerized PABX: an Electronic Office. Integrated Office Systems-Burotics, ed. Naffah, North-Holland, 1980.

71. Scheurer, B. Office Workstation Design. Workshop on OIS, St-Maximin, October 1981, Republished in: Naffah & al., Office Information System, North-Holland 1982:105-116.

72. Sirbu, M. Understanding Managerial Group Work. Proceedings of the 3rd AFIPS Office Automation Conference, April 1982.

73. Sirbu, M.; et al. OAM: An Office Analysis Methodology. MIT Laboratory for Computer Science, Office Automation Group Memo, Cambridge, Massachusetts, 1981.

74. Smith, D.C.; Harslem, E.; Irby, C.; Kimball, R. The Star User Interface: An Overview. Proceedings of the National Computer Conference; 1982 June; Houston, 515-528.

75. Smith, S.A.; Benjamin, R. Projecting Demand for Electronic Communications in Automated Offices. ACM Transactions on Office Information Systems. 1983 January; 1 (1).

76. Suchman, L. Office Procedure as Practical Action: Models of Work and System Design. ACM Transactions on Office Information Systems, 1983 January; 1 (4).

77. Tapscott, D. Office Automation: A User-Driven Method. Plenum Press, 1982.

78. Thurber, K.J. A Survey of Local Network Hardware. Proceedings of COMPCON Fall 80, September 1980, 273-275.

79. Tilbrook, D. Information Systems Primitives. Proceedings of the International Workshop on Integrated Office Systems, November 1979.

80. Tsichritzis, D.; Rabitti, F.; Gibbs, S.; Nierstrasz, O.; Hogg, J. A System for Managing Structured Messages. IEEE Transactions on Communications, 1982; Vol. Com-30 (1): 66-73.

81. Tsichritzis, D.C. Integrating Data Base and Message Systems. Proceedings of the 7th International Conference on Very Large Data Bases, 1981. 356-362.

82. Tsichritzis, D.C.; Christodoulakis, S. Message Files. ACM Transactions on Office Information Systems, 1/1:88-98, 1983. Also presented at the ACM SIGOA Conference on Office Information Systems, Philadelphia, 1982.

83. Turoff, M. Information, Value, and the Marketplace. New Jersey Institute of Technology Technical Report, September, 1983.

84. Wegmann, A. VITRAIL; A Window Manager for an Office Information System. Proceedings of the Second ACM-SIGOA Conference on Office Information Systems; 1984, June 25-27: Toronto, New York: ACM; 1984; 2:1.

85. Williams, G. The Lisa Computer System. BYTE 8/2:33-50, 1983.

86. Zdonik, S.B. Object Management System Concepts: Supporting Integrated Office Workstation Applications. Ph.D., Thesis, Massachusetts Institute of Technology, May, 1983.

87. Zimmermann, H. OSI Reference Model-The ISO Model of Architecture for Open Systems Interconnection. IEEE Transactions on Communications, April 1980, 425-432.

88. Zisman, M. Representation, Specification and Automation of Office Procedures. Ph.D. Thesis, Department of Decision Science, The Wharton School, University of Pennsylvania, Philadelphia, Pa., 1977.

89. Zloof, M.M. QBE/OBE: A Language for Office and Business Automation. IEEE Computer, 14/5: May 1981. 13-22.

90. Zloof, M. M. Office by Example: A Business Language That Unifies Data and Word Processing and Electronic Mail. IBM Systems Journal. 1982; 21 (3): 272-304.

ARABIC INPUT/OUTPUT TO COMPUTERS

A Survey of Bilingual Peripherals

N. HARFOUCH
SSRC
P.O. Box 4470
Damascus, Syria

S. KOTOB
KISR
P.O. Box 24885
Safat, Kuwait

1. INTRODUCTION

In recent years there has been a steadily growing interest in computer based applications in the Arab World. The reasons behind this lies in the state of development within the Arab World, as well as in the specific trends of information technology, and industry in the rest of the world.

A detailed consideration of these reasons would be beyond the scope of this presentation. It is interesting to note, however, that developments in the areas of semi-conductor components and electronic systems technologies have been, and still are constantly reducing the size, weight, and cost of computer equipment, while increasing computer power, and the range of tools and applications available to the end-user. It has become clear, that the new barrier against the spread of modern computer-based information technologies in developing societies is the depth of computer literacy, which depends particularly on the ability of computer systems to interface with the users in their natural languages.

The perspective of a huge potential market, estimated to be in the billions of dollars, which could result from successfully crossing that barrier, has led to the accelerated development of arabized computer hardware and software which the field has witnessed in the recent past.

The first attempts in arabization were made by large multinational corporations. They were ill-fated as they were made with a background of centralized second generation machines for classical white-collar computer center applications, and hence gave priority to the already existing product range of each corporation over the necessity of solving the deep inherent problems of arabization in order to advance towards the complex multi-vendor networks of the forseeable future.

The second generation of arabized equipment was triggered by technological developments, and the first timid attempts to standardization in the Arab World.

The possibility of adding considerable local intelligence to peripherals through microprocessors, the advances in semi conductor memory allowing multiple fonts, software generated character sets and keyboard tables, and the emerging interest in standardization within the Arab World, encouraged the appearance of scores of small companies in North America and Europe dedicated to the arabization of equipment and applications. These companies were willing to undertake a more complex approach to arabization in order to capture any share of this promising market through products exhibiting more added value.

They were closely followed by the large manufacturers, who undertook simultaneously a dialog with the emerging regional and national Arab standardization organizations.

It is interesting to note at this stage that three main bodies are normally involved in any similar development process. Those are the manufacturers, the users, and the standardization organization. It would seem today that in the area of arabic/latin computer systems the manufactures are well established. In addition to the small dedicated companies mentioned above, almost every major manufacturer of computer systems has arabic capability in his offerings or plans, creating a competition which encourages further developments.

It would seem also that the Arab Standardizations Metrology Organization (ASMO) is now universally accepted as the coordinating body for Arab standardization organizations and as the major body for studying, developing, issuing, and promoting the standards related to computer technology in the Arab World.

It must be noted however, that only very few

This work was partially funded and carried by Kuwait Institute for Scientific Research as a part of the Arab Computing Project (ASD-7), and by the Scientific Studies and Research Center (Syria), as part of the Arabic Intelligent Terminal Project (AP09).

computer user groups exist in Arab countries. Despite all recent developments, the effort to bring computing to Arab offices, schools and homes, is still very fragmented, and of limited success. This is due partly to the fact that implementing Arabic on Latin hardware, which was not initially conceived with the special requirements of Arabic in mind, is difficult, and partly because basic research and development that deals with the special requirements of Arabic is scarce. The abscence of such R&D support in ASMO as well as the lack of any systematic feedback from users has rendered the urgently needed work of standardization slow and sometimes more disruptive than constructive to the development process.

In a domain driven mainly by lucrative technological breakthrough and high profits, large computing companies, both hardware manufactuers and software houses, have failed to allocate the necessary funds for basic R&D in Arabic computing, while small companies have been financially constrained from such undertaking.

Measured by the experience of other countries, especially Japan and India, it appears that the basic issues of Arabic computing can only be solved by a dedicated team of scientists and engineers with a wide range of experience in computer science, Arabic linguistics and engineering. Because of the constraints mentioned above, such a team can be assembled only by non-profit organizations that understand long-term benefits, which cannot be measured in direct monetary values. Such a team could very well be assembled by one or several research institutes in the Arab World.

It is against this background that work was initiated independently in both the Kuwait Institute for Scientific Research (KISR) and the Syrian Scientific Studies and Research Center (SSRC), in an attempt to define a long-term Arabic computing program.

The following over-all objectives were common to both works:

1. Establish a viable national capability in computing in Arabic.
2. Establish an area of expertise within each institution in support of Arabic computing and its applications, that can aid efforts to standardize methods and techniques for Arabic computing nationally and regionally.
3. Develop seed for application software in selected areas.
4. Define requirements, guidelines, and standards to develop computer equipment specifically designed for use in an Arabic environment.

The present paper summarizes a part of the results of both [1] and [2], mainly related to text input/output in Arabic through bilingual peripherals presently available.

2. HARDWARE

The assessment phase consisted of analysis and surveys of hardware and software available to both teams, and of user surveys and pilot developments aimed at drawing a representative picture of the present status of Arabic computing .

The assessment was not intended to provide an endorsement for any particular hardware, system or application software, but to contribute to the definition of requirements for Arabizing existing hardware, operating systems, programming languages, and applications for the development of fully Arabized computer equipment.

Evaluation guidelines were developed for each category of assessed equipment and systems. Because of the slightly different orientation of KISR and SSRC teams and because the projects were initiated independently, the emphasis in the two projects was not exactly the same. Both projects assessed arabized microcomputers in detail. But while the KISR team concentrated in the first phase on all aspects of arabized microcomputers, including operating systems, programming languages and applications, the SSRC team placed the emphasis of its work on hardware related issues and investigated arabized microcomputers, terminals, and printers.

For the sake of consistency, and in order to remain within the space allocated to this presentation, the synthesis of the results of both efforts had to limit itself to those assessments made according to identical or similar evaluation guidelines.

2.1. Microcomputers:

The microcomputers selected for assessement were the following:

- o National/MSX
- o Apple II + and Apple IIc microcomputers
- o IBM-PC, Gulf Computing Company (GCS), Arabic board
- o Victor 9000 (previously Sirius-1)
- o NCR Decision Mate V (DMV)
- o Falcon 2500 (manufactured by Falcon Systems Ltd)
- o Al-Raed (manufactured by Research Computer Technology Co.)
- o Al-Farabi (made by Saudi Computer Industries originally selected for assessment but was unavailable).

Both teams were aware of many fine hardware microcomputing systems with relevant innovative features to Arabic computing, as well as several announcements of newly arabized systems, which were not available to them in the time frame in which the assessments were carried out. The list of assessed hardware is therefor by no means comprehensive, but it is believed to be representative.

The MSX was chosen for a home computer, with a high potential for penetrating the home and educational computer market, as the MSX standard, made by the well known Microsoft, has been adopted by more than 16 large companies, mostly Japenese. Recently Al-Alamiah Electronics of Kuwait and Microsoft announced the Arabized MSX named MSX صخر .

The Apple II family was chosen as a second 8-bit machine possessing both home and personal

computer features and most widely used for education.

For personal/business microcomputers several machines representative of the wide range of available hardware were selected:
- o The 8/16 bit IBM-PC widely used and with many compatibles in use.
- o The Victor 9000 using the same 8/16 microprocessor as the IBM-PC but with a different architecture and several innovative features of interest to Arabization (software loaded character fonts and keyboard tables very high resolution display etc...) is interesting for the evaluation of the impact of the architecture and the said features on arabization.
- o Both the NCR DMV and the Falcon 2500 systems represent more recent entries in the upper range of business microcomputers with an 8-bit and a 16-bit microprocessors and multiple operating systems.

The first represents the entry of a well established large computer manufacturer, offering the possibility of assessing the over-all systems compatibility between different generations of such manufacturer's equipments, and the degree to which the new strandards have penetrated its product range.

The second represents the entry of a small second generation company, dedicated to arabization, and offers in addition several interesting features such as a concurrent operating system, multiple serial ports for the connection of up to 6 terminals, a fully bilingual word processor, and a patchable format floppy disk drive, all in the size of a single user desk-top microcomputer.

Al Raed, finally, was to the best knowledge of the authors the only fully Arabized microcomputer available during the project period.

2.2. Bilingual Terminals:

Although the microcomputer is ideally suited for stand-alone applications, and hence lends itself to widespread applications, and achieves such mass significance, it should not be overlooked that there is an important layer of applications based on minicomputers and their clusters of terminals. The minicomputers have been subjected to the same trends of technology which have influenced microcomputers so strongly.

On the one hand, the reduction in sizes in all varieties of computers and the increased performance and memory capacity they display, are making the boundaries between minicomputers and both microcomputers and mainframes blurred . On the other hand, it is well established today that stand-alone microcomputers sooner or later necessitate communication with larger resources, which is usually implemented through software emulating in the microcomputer the behaviour of terminals connected to host. It was felt therefore, that Arabic terminals have a relevance in the assessement of Arabic peripherals, in particular for certain educational and business applications.

The selected terminals were all chosen from the second generation of Arabic equipment. They include:
- o Alis 970
- o Comterm 541
- o NCR 7910
- o Integro DKU 7140S

2.3. Printers:

Applications based on stand-alone microcomputers, or on clusters of terminals, both necessitate generally a printing device for output.

This study considered dot matrix printers capable of hard-copying bilingual terminals, and/or of stand_alone operation (serial connection to a host):
- o Alis 3304
- o Comterm 548
- o Integro APS 180

3. EVALUATION GUIDLINES

Guidelines were developed in order to assess the arabization of the selected microcomputers, and to identify the relevant hardware features involved in the arabization.

Processing hardware details were studied and compared including the type of microprocessor used, the maximum available RAM and ROM and the mass storage devices attached to the processor.

The microprocessor determines to a certain extent the operating system offered, and this again has an influence on the way the arabization is carried out.

The available amount of RAM and ROM determines the available degree of freedom in the implementation of the additional software and/or firmware required for Arabic.

Both KISR and SSRC projects included other tasks than the survey and assessement of Arabic text input/output, and work included a study and review of other aspects of microcomputers, such as arabized operating systems, programming languages and applications software. Some of the results of these other tasks, while primarely intended for an assessement of Arabic computing, were included here as background describing the general environment in which the text input/output features are embeded.

Input/output features are of special interest for arabized systems. They were considered in greater detail. Parameters studied included the resolution offered by the visual display unit (VDU), the size of the matrix and cell used for generating the Arabic characters, the character shapes achieved in the different implementations, the code used for the representation of the Arabic characters, the handling of numerals, and the availability of color display and graphics.

Similarly the display characteristics of VDUs were assessed, in particular the trade-off between the size of matrix and the shape of

characters under several assumptions, including full (simple and double) representation of diacritics and the use of ASMO-449 or other codes.

The Arabic and/or bilingual keyboards offered with the considered microcomputers were evaluated in common with the surveyed bilingual terminals. Features considered included the number of keys available, the number of function keys, the physical design of the keyboard, and the layout of the Arabic characters.

In view of the lack of approved standards for Arabic keyboard layout, it was necessary to develop a reference against which the surveyed keyboards could be evaluated. This reference layout was derived from the common parts of two recents proposals for such a standard presented by the European Computer Manufacturers Association (ECMA) [3] and ASMO [4].

Originally ASMO's proposal was based on the occurence frequencies of the characters, but it evolved later towards more compatibility with the typewriter keyboard layout.

Table 1 shows the characters which have the same positions in both proposals and those which have different positions in each proposal. Figure 1 shows the reference layout for this character statistics approach. A second reference was devised from the normal electric typewriter keyboards. Figure 2 shows the reference layout for the typewriter approach.

Another important issue in the Arabic text input/output area is contextual analysis, (CA), a basic characteristic of which is now second generation Arabic equipment. In order to assess the many versions of CA algorithms, a reference model was devised consisting of ten levels beginning with the fundamental features of CA. and featuring more

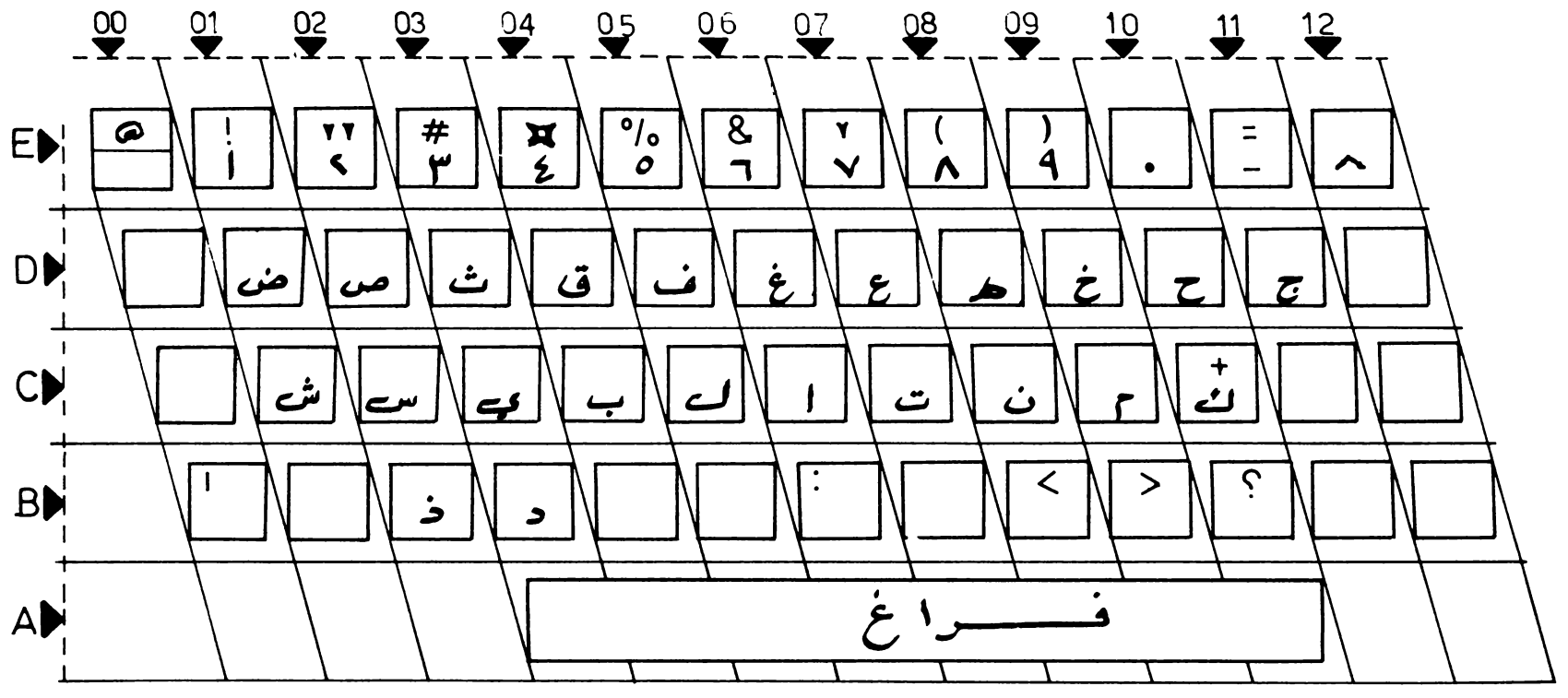


FIGURE 1. The statistical reference layout

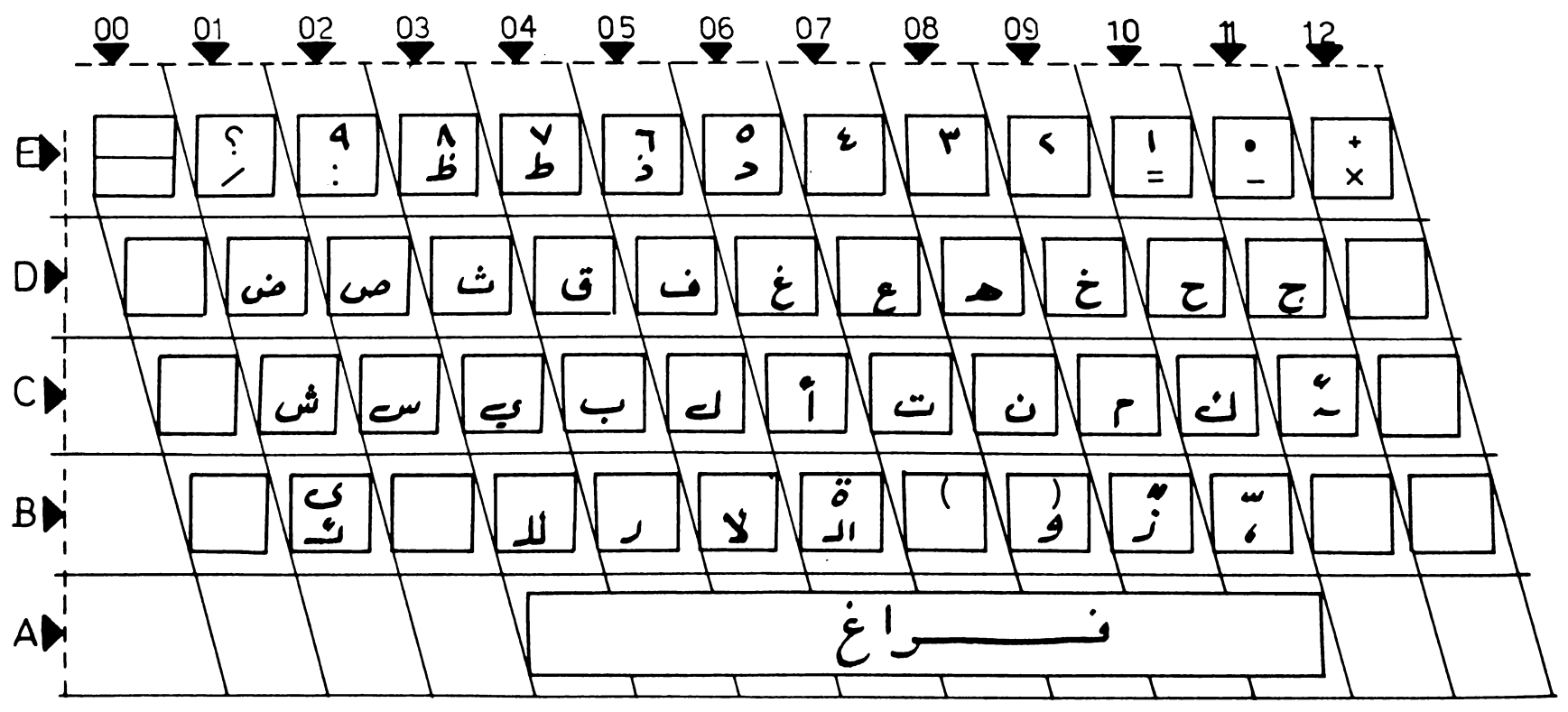


FIGURE 2. The typwriter reference layout

Table 1. Summary of key assignment proposals

TWR	ECMA	ASMO	ASMO & ECMA	ch.
uC11	B05	B06		ء
N/A	uC05	uC12		آ
N/A	uC06	C12		أ
N/A	B07	uB07		ؤ
N/A	uB04	uC05		إ
B01	B01	C11		ئ
			C06	ا
			C04	ب
uB06	B06	uC07		ة
=			C07	ت
=			D03	ث
=			D11	ج
=			D10	ح
=			D09	خ
=			D03	د
=			D02	ذ
B04	B04	B05		ر
B09	B09	B04		ز
=			C02	س
=			C01	ش
=			D02	ص
=			D01	ض
E04	C11	B01		ط
E03	C12	B00		ظ
=			D07	ع
=			D06	غ
=			D05	ف
=			D04	ق
=			C10	ك
=			C05	ل
=			C09	م
=			C08	ن
=			D08	ه
B08	B08	B07		و
uB01	B00	uC08		ى
			C03	ي
=			A	فراغ
N/A	uD05	uD03		ّ
N/A	uD04	uC02		ً
N/A	uD03	uB01		ٌ
N/A	uD06	uD04		ٍ
N/A	uD07	uD03		َ
N/A	uD08	uB02		ُ
uB10	uC07	uD06		ِ
N/A	uD09	uD05		ْ

= means same as ASMO & ECMA

sophisticated (and less critical) features with growing level numbers. The concept of levels of CA was proposed earlier by Mackay [5], before the publishing of the ASMO standard 449, and hence included processing features, which would not be compatible with it. The proposal consisted of seven levels. This concept was further developed to match ASMO 449, and to adapt to technological improvements in the equipment. Table 2 describes the functions assigned to each of the ten proposed levels.

The last part of this survey is concerned

Table 2. Evaluation levels of contextual analysis

Level	Function
1	Differentiation between initial/middle and final/independent forms of characters
2	Correct handling of numerals
3	Correct handling of all combinations of Hamza with its carrying characters (ء آ أ ؤ إ ئ ٱ ٵ)
4	Correct handling of the textual/decimal seperator and
5	Correct handling of all simple diacritics (ً ٌ ٍ َ ُ ِ ّ ْ)
6	Correct handling of and sequences, even when a special ﻻ or ﻷ is supported
7	Differentiation between all four character forms (initial/middle/final/independent)
8	Correct handling of double diacritics (combinations of ّ with other diacritics)
9	Correct handling of characters in final and/or independent positions which are extended with connecting dash
10	Transformation of ه in initial or middle positions into ـه

with second generation bilingual printers. The use of microprocessors in the control unit of these printers enables the implementation of contextual analysis features within the printers. Parameters and features surveyed included general technical specifications, available character sets, fonts, printing modes (Letter, Near letter, Draft and EDP quality), printing speeds achieved, size of matrix used in each mode, printing attributes, and the graphics capability.

4. SURVEY RESULTS

4.1. Microcomputers:

Table 3 summarizes the results of the microcomputer survey. It appears from this survey that the Arabization process is penetrating rapidly the complete range of available microcomputers, from small 8-bit microcomputers for home and educational applications to powerful multi-user systems for business applications.

Furthermore first attempts at designing completly Arabic microcomputers are already at hand. In addition to Al-Raed computer, reviewed in this survey, a second fully Arabic system, Al-Farabi was announced but not available at the time. While these

Table 3. Arabic microcomputer survey summary

FUNCTION \ SYSTEM	MSX FAMILY	APPLE II FAMILY PATTY ACCESS	IBM-PC (GCS)(2)	VICTOR 9000 (ARABRITE)	NCR DMV	FALCON 2500	AL-RAED 100
PROCESSING							
Processor	8 bit	8 bit	8/16 bit	8/16 bit	8+16 bit	8+16 bit	8 bit
Memory capacity	64 K	128 K	640 K	896 K	512 K	789 K	280 K
Cassette tape	A	A	A	N/A	N/A	N/A	N/A
Cartridge	A	N/A	N/A	N/A	N/A	N/A	N/A
Floppy diskettes	A	A	A	A	A	A	A
Hard disk	N/A	A(1)	A(1)	A	A	A	N/A
INPUT/OUTPUT							
Arabic Display							
Matrix size	2 sizes (8x8)	2 sizes	9x14	13x16	7x19(4)	9x13	8x20
Character shape	G	OK	G	G	G	G	VG
Code used	ASMO	ASMO	OWN	OWN	OWN	ASMO	OWN
Color display	A	Possible	N/A	Possible	A	A	N/A
Graphics + Arabic	A	A	N/A	A	A	A	N/A
Resolution	Low/Medium	Low/Med	High	Very high	High	Opt(High)	High
OPERATING SYSTEM							
Types available	MSX	DOS+CP/M	MS-DOS	MS-DOS CP/M86 UNIX	CP/M/86 CP/M-80 MS-DOS UCSD P	CPM86/80+ CCPM86/80 MS-DOS	Arabic OS: (CP/M)
Arabic extension	MSX SAKHR			ARAB DOS ARAB SYS	Arab CP/M-80 Arab MS-DOS		AL-RAED
Arabic commands	N	N	N	N	N	N	Y
Arabic error message	N	N	N	N	N	N	Y
Commands with Arabic files							
ARABIC PROGRAMMING LANGUAGES	BASIC(3)	BASIC(3) PASCAL(3)	BASIC(3)	BASIC(3)	BASIC(3) COBOL(3)	BASIC(3)	AL-KHAWA-RIZMI
ARABIC APPLICATIONS SOFTWARE							
Bilingual WPS/W	Y	N/A	Y	Y	Y	Y	Y
Access Arabic files	-	-	Y	N	N	Y	Y
Access bilingual files	-	-	N	N	N	Y	Y
Accounting	N/A	N/A	N/A	N/A	Y	Y	Y
Spread sheet	N/A	N/A	Y	N/A	N/A	N/A	Y
DBMS	N/A	N/A	Y	Y	N/A	N/A	Y
Education	Y	N/A	N/A	N/A	N/A	N/A	N/A
Scientific	N/A	N/A	N/A	N/A	N/A	N/A	-
Voice Synthesis	N/A	N/A	N/A	Possible	N/A	N/A	-
Automatic Translation	N/A	N/A	N/A	N/A	N/A	N/A	-
Communication	N/A	N/A	N/A	N/A	N/A	Y	-
GENERAL							
Price	Low	Medium	High	High	High	Medium	High
Manufacturer support	Unknown yet	Poor	Poor	Poor	Medium	Medium	-
Physical portability	Medium	N/A	N/A	N/A	N/A	N/A	N/A Very heavy

1. Available from other suppliers
2. Prior to the publication of KISR report, IBM announced the Arabic IBM-PC
3. Insertion method
4. Cell size used on NCR 7910 in the DMV mode. Indication was given only that the Arabic character occupies a larger cell than that of the English character [12]

efforts towards fully arabized microcomputers should be applauded and represent a more fundamental approach to Arabic microcomputers, several points must be stressed:

1. The motivation for a radically Arabic oriented hardware and software design was and still is the unavailability of certain

features, which are important for Arabic and bilingual operations in the present microcomputers, as these were originally designed to handle non-Arabic (mostly Latin) operations.

It should be noted however that in the upper range of the equipment, more and more powerful microcomputers are appearing, which are gradually offering some of these desired features. This development is not matched at the lower end of the range, despite some obvious advances and improvements. It should be therefore carefully considered, whether the design of "fully Arabic" microcomputers should not perhaps concentrate on that end where obtaining the desired features is more difficult. The main obstacles are the hardware/software limitations of small systems and price constraint on medium systems if they are to be affordable to users in home and education applications. It is also the case that at this end bilingualism is less important and the unavailability of Arabic is more the main barrier to wide spread use of microcomputers in the Arabic society.

2. If these commendable efforts are to have a durable impact on Arabic computing, care should be taken to permanently preserve a dual compatibility: one between Arabic microcomputers, which would be beneficial to manufacturers, software developers, and users, and a compatibility (through continuous R & D) with the rapidly changing state of the art in computer science, engineering and technology. It is regretable that the two Arabic systems existing today have shown no compatibility between themselves at any level of software of hardware.
It is encouraging on the other hand, that new developments are pursued by the Arabic computers designers/manufacturers. One KISR researcher was able to inspect a Motorola 68000 based system, a further development of Al-Raed range, in the last stages of prototyping. When introduced it should be superior to Al-Raed 100, and comparable to systems recently announced by major micro manufacturers.

The most distinctive characteristic of the arabized microcomputers, regardless of their class, is certainly their non-compatibility with one another in all respects of arabization. In the developed world, some industry standards have been generally accepted even before being officially endorsed. Despite the high competitiveness of the international computer market, ASCII for example is a well established, understood, and accepted term even with more than eight different national versions. CP/M, MS-DOS and Unix, the major operating systems of microcomputers are trying to establish minimum commonality between each other (the latter perhaps less than the former two), and each is offering emulations of the other systems in order to increase the number of application software available to their users.

Due to the lack of standardization for Arabic computing issues, and to insufficient understanding of these issues, the different implementations of the Arabic extensions to major operating systems (such as MS-DOS and CP/M) are imcompatible with each other. They differ in the method of switching from one language to the other, in their Arabic numeral-handling conventions in the injection of one language into the other in the implementation of the contextual analysis, and even in the Arabic code used, despite the existence of an official standard (ASMO-449) for such a code. This situation has further narrowed the offerings in application software for Arabized computers and hence made these less attractive for the new development of entertainement, education and business applications.

A further result of this survey is the absence of sufficient arabized or Arabic application software. This is partly related to the previously discussed issue of non-standardization of the Arabized and Arabic operating systems, and also to the general state of Arabic computing in the present.

The bulk of applications developed are word processing packages usually called "bilingual". The majority of these word processors operate in one of two language mode, each mode supporting injection of relatively short sequences of the other language. Perparing a long and fully bilingual document, such as a bilingual contract can be a very tedious job, because of the limitations of processing in the injection mode. These word processors should better be designated as "dual language" word processors. Few other packages support fully bilingual operations, including setting up different language zones in the same document, and converting automatically from one mode to the other as soon as a certain zone is entered. This would give the full range of processing features of both language modes in the same document and practically simultaneously.

The next areas of applications are financial and accounting applications, including spread sheet programs, and data base management systems. These programs are usually offered independently.

Recently, there has been a move toward offering integrated microcomputer based packages for Arabic office automation [6][7]

Educational software is just being introduced. Scientific applications are almost non-existent, but it is hoped that experimental developments similar to the OMRAN system [8] will lead to mature products in the near future.

In the absence of a minimum of standardization, the entire class of communication packages for the emulation of bilingual terminals or the transfer of files is scarce and limited to within the product range of each vendor if any.

4.2. Terminals:

The results of the survey of bilingual

Table 4. Summary of terminal assessment

FUNCTION \ TERMINAL	INTEGRO DKU 7140S	ALIS 970	COMTERM 541	NCR 7910
Self diagnostics	Yes	Yes	Yes	Yes
Comm. interface	V. 24 and V. 11	V.24, 20mA loop	V.24 (RS232)	V.24 (RS232)
Printer interface	Yes	Yes	Yes	Yes
Hard copy printer	APS 180	Alis 3304	Comterm 548	NCR 6411
Display: CRT	15'	14'	15'	15'
Matrix Latin		6x8	8x12	
Arabic	10x15	8x10	10x24	7x9(5x9)(1)
Cell	10x15	8x10	10x24	7x19(6x19)(1)
Attributes	Normal-underline Inverse-blank Low int-blink Column seperator	Normal-underline Inverse-blank 1/2 int-blink DH - DW	Normal Inverse-blank High int-blink	Normal-underline Inverse-blank Intensify-blink DH - DW - bold
Shape of characters	Good	Medium +	Very good	Fair
Codes Available	ASMO 449 Comterm 341	ASMO 449, ARCII CODAR-UFD, SASO Comterm 341, 541	Comterm	NCR standard NCR non-standard (both non-ASMO)
Character sets	Arabic English or French	Arabic English or 1 of 9 other languages	Arabic English (upper case) Optional: English (lower case)	Arabic English
Modes available	Teletype model 33 model 35	VT 100 VT 52	HP protocols	NCR 7900 model 1 NCR DMV
Display capacity	24x80 1 status line	24x80 or 132 1 status	24x80 1 status	24x80 or 132 1 status
Number of shapes	255	512	256	?
Graphics	No	VT100 special set Opt. tektronix 4010/4014	No	No
Auto dimming(2)	No	5, 10, 15 minutes	10 minutes	5 minutes
CA level				
Keyboard: Design	Detachable	Detachable	Detachable	Detachable
Number of keys	100	141	97	108
Desk space required	3024 cm^2	1710 cm^2	3035 cm^2	2596 cm^2
Relative price(3)	1.31	1	1.2	1.6

1. For 132 character/line
2. Automatic dimming of CRT on inactivity
3. Relative to VT 100 price
   DH = Double height
   DW = Double width

terminals (Table 4) has shown, that there are already some detectable trends in this area, but that the necessary standardization has again not yet reached a satisfactory level.

Common hardware features are:
- o Microprocessor control
- o Buffering
- o Large character display matrix
- o One or more communications interfaces
- o Interface to some sort of hard copy printer
- o Firmware-based menu-driven configuration of terminal
- o Large character generators to accommodate both Latin and Arabic character sets
- o Multiple Arabic codes available
- o Rather large physical size
- o Detachable keyboards with reduced character set
- o Contextual analysis
- o 30-40% higher priced than a standard ASCII terminal (such as the VT100 for example).

While the availability of several code sets is evidence of the insufficient standardization, it should be kept in mind that the majority of existing Arabic applications have been written in non-ASMO codes. The conversion of all these applications to ASMO code is not practical and the approach generally taken is to implement new applications in ASMO code, while keeping previous ones as they are and phasing them out gradually. Such a transition could extend over years and implies terminals capable of using several codes, and switching easily among them.

The display. One of the difficulties that no doubt faces any arabization is the presentation and display of Arabic characters on video screens, because of the calligraphic nature of these characters. It seems well established now, that if an acceptable calligraphic quality is to be achieved,while simultaneously respecting the required 1-1-1 relationship between the character representation on the keyboard, in the code

and on the display, a large character display matrix must be used.

The results of both teams concerning the required matrix size appear to be converging: 9x12 and 9x14. This includes the presentation of diacritics along the same line as the characters as implied by ASMO 449. The achievable quality with such a matrix size is quite acceptable and deviations from the normal Arabic font only becomes noticeable when using double diacritics with a character (Shadda combinations with other diacritics). If desired, the shape of characters possessing a cup in their final and independent forms might be improved by allowing these characters to use a part of the space following. This could be implemented by allocating several CA dependent shapes to the space character.

If for any reason it becomes necessary to present diacritics above and below characters (with all the obvious implications of such a departure from the standard) two techniques can be used. Diacritics can either be placed on separate lines above and below the text line (as practiced by IBM/Kuwait Scientific Center for example), or they can be implemented in the same matrix (as in the case of Al-Raed 100).

In the first case, each line of text would occupy three lines on the display and in the memory, hence limiting the amount of text displayed on the screen. This limitation can be compensated by using the ZOOM function, allowing the user at the depression of a key, to display the full text page without its diacritics or alternatively to zoom-in on a part of that page determined by the cursor position and display that part with its diacritics.

In the second case a larger character matrix is required. Al-Raed uses an 8x20 matrix. A matrix of 10x24 would be necessary for the high quality presentation of double diacritics and the character in the same matrix.

Another approach worth considering is the variable width character matrix (such as that adopted by Al-Alamieh Electronics in Kuwait). In this approach, the shape of the character decides the width of the dot matrix implying that diacritics can be placed closer to the character itself. The result is very close to written Arabic, but this approach is more complex to handle.

Contextual analysis. CA in Arabic and Arabized terminals and microcomputers is today a standard feature, although the quality and the performance of the CA algorithms used vary widely.

The way in which some functions of CA are implemented is related to the Arabic code used. An effort was made therefore to keep the CA functions defined in Table 2 as independent from the code set as possible, although the relevance of certain functions to a particular code set could vary as a result. Function 4 for example is less critical to the Comterm code set, as this code defines two distinct characters as textual and decimal separators, than to the ASMO code, where both separators are defined as the same character and the significance and shape in either environment are defined through CA. The same applies to the various degrees of diacritics support provided by each equipment.

The results of the CA evaluation (shown in Table 5) indicate that in some instances very fundamental functions such as the correct handling of numerals in Arabic (function 2 in Tables 2 and 5), are not fully implemented. In general the Lam/Alif sequence is not processed correctly, in particular when a special Lamalif key (generating one or two codes depending on the used code) is provided on the keyboard. The presence of such a key is not enough justification for not providing this function, as a CA should not allow the presentation of non-existent character shapes.

Table 5. Summary of CA evaluation

CA LEVEL	DKU 7140	ALIS 9700	COMTERM 5410	NCR 7910	FALCON WRITER	ARABRITE	AL-MOUHKAM	NCR FORWARD
1	F	F	F	F	F	F	F	F
2	F	F	F	F	P	P	F	F
3	F[2]	F	F	F	F	N	N	F
4	P	F	F	P	P	N	N	P
5	F	F[1]	N	N	F	P	P	N
6	F	F	N	N	N	N	N	N
7	F	F	F	P	P	P	F	P
8	F	F[1]	P	N	F	N	P	N
9	F	N	N	N	N	N	N	N
10	N	N	N	N	N	N	N	N

F = Full, P = Partly, N = Not available
1. Non ASMO
2. Two-key sequence

The differentiation between initial and middle, and between independent and final forms of the character is provided in various degrees on all equipment. These degrees range from only three characters to almost all characters. In the latter case the results of such wide differentiation were not commensurate with the effort; it was found that a more careful design of the character shapes could noticeably improve the quality of the display while reducing the total number of necessary shapes.

The correct CA presentation of characters in independent and final forms, when they have been extended by one or more connecting dashes, is generally not provided, although such extentions represent a widely used and language specific method for margin justification and creation of titles in Arabic. Function 10 was included in an attempt to find out whether any advanced features are being added to the CA algorithms towards improving the detection of basic misspelling errors. Such features have not been identified in the surveyed equipment.

The keyboard. Similarly to the display, the keyboard issue is common to both microcomputers and terminals. Proposals for the standardization of the keyboard layout have been discussed since 1978. These proposals have been traditionally grouped in two main categories: layouts based on the usual typewriter keyboard, and layouts (more or less) based on the occurence frequencies of the characters.

The reference models for the evaluations of today's Arabic keyboards in this survey were chosen accordingly as explained in 3. The results of this survey are summarized in Table 6. Common features in all surveyed keyboards are:

- o Detachable design
- o Programmable function keys
- o Numerical pad
- o Additional composite keys

Both layout types are almost equally represented in the surveyed keyboards with a slight advantage for the "typewriter" layout. The conformity with the reference model within this group is generally high, because the classical typewriter keyboard is better defined and established.

A closer look at both reference layouts (Figures 1 and 2) shows that the classical typewriter keyboard has evolved under the influence of CA in the direction of the former "statistical" keyboard. In fact two rows of both layouts are identical (D01 to D11 and C01 to C10) and the problem has changed from the selection of the fundamental design philosophy of the keyboard towards the optimization of an existing draft design.

The main principles applied to this design are the following:

1. It is based on ISO 2530 standard, allowing for 48 graphic character keys.
2. It allocates all alphabetical Arabic graphic characters to keys in rows B, C and D for ease of use.
3. It will allocate the maximum number of unshifted positions to the graphical characters.
4. It deviates from the traditional typewriter layout only to the absolute minimum, in order to eliminate costly retraining.
5. It will allocate the most frequently used characters to the most accessible keys.
6. It will group families of characters on adjacent keys in order to improve visual mnemonics.
7. Its latin part will not be identical to any national or international standards, but it will correspond closely to the layouts used in the Mediterranean and the Middle East.

Areas of difference are the following:

1. If the diacritics are considered as integral part of the character set as some proposals do, then the application of the third design principle will lead to allocating part of the diacritics to unshifted positions, to the detriment of the mnemonic grouping considered under design principle 6. Because no diacritics were supported on the classical typewriter layout (except for) compatibility with older keyboards of this type seems not critical.

Table 6. Summary of keyboard evaluation

Characteristics \ Keyboard of	Arabic IBM-PC	NCR DMV	Victor Al-Mouhkam	Victor Arabrite	Falcon Writer 1	Alis 970	Integro DKU 7140S	NCR 7910
Number of keys (total)	83	99	99	99	109	141	100	108
Number of function keys	10	20	10	10	32	16	12	10
Numerical pad keys	13	10	17	17	11	14	14	13
Reference type (STA or TWR)	STA	STA	TWR	TWR	TWR	STA	TWR	STA
Conformity with reference	71.7	77.4	50.9	96.6	84.7	77.4	100	62.7
Additional Alphabet keys	3	2	4	3	1	7	5	2+(4)*
Diacritics layout	N/A	N/A	own	(subset only)own	own	own	own	N/A

* Non-Arabic characters

2. Design principle 5 assumes knowledge of the character frequencies. The studies available so far on the frequencies are still controversial.
The study made by Iraq's NCC [9] considered 565,981 alphabetical characters with all Hamza forms. The study made by the moroccan IERA [9] considered 1,300,000 characters including the space and diacritics characters. The Syrian SSRC considered in a recent study a sample of 4,869,000 characters (without space) taken from a typical EDP applications input/output environment. The sample included 1,755,000 alphabetical characters. A comparison of these results and comments on their relevance to particular areas is given in [10]. Neither the frequencies obtained nor the sequence of characters when ordered according to their frequencies converged sufficiently. Considering that 21 positions of the keyboard are agreed upon and practically accepted, as this survey has shown, efforts should be oriented principally towards the correct assignment of the remaining alphabetical characters according to their established sequence of frequencies to the unassigned keys. These characters are:

   and hamza combinations

3. The accessability of a certain key on the keyboard is assessed also in respect of the time necessary to reach that key. The initial positions of the fingers on the keyboard also influence these access times [11]. Some proposals have based their optimization process on initial positions which are on rows B and C and not only on B, which is the more usual convention. A survey of the actual initial positions used by typing personnel should confirm which of the initial positions is the one to be retained and used for the optimization of the unassigned characters mentioned above.

Considering the urgency of establishing standards and the pressure resulting from more products brought to the market before the standards are established, it appears that only one of two possible approaches can be adopted: either achieve rapid agreement between the concerned parties, or settle for two different keyboards, for word processing and for EDP, as modern technology permits to making both appear identical to the system they are attached to.

4.3 Printers:

The survey of printers has been limited to the dot-matrix type for their following characteristics:

- Widely used with microcomputers and terminals
- Low cost
- Bilingual printing without change of hardware or use of sophisticated mechanisms (e.g. dual daisy wheel)
- Matrix flexibility for representation of arabic characters including some printing attributes (enlarged characters, small characters etc...)
- Capability of economic draft quality and near letter quality printing

The survey selected printers offering communications interfaces as well as hard copy support for surveyed terminals. Table 7 summarizes the survey results. Common features of today arabized dot matrix

Table 7. Summary of printers evaluation

Function \ System	Alis 3304	Comterm 548	Integro APS 180
Buffer memory	512 Character	2048 Character	2048 Character
Comm. interface	RS 232(other optional)	RS 232 (for hard copy)	V.24 or IEEE 488
Hard copy terminal	Alis 970	Comterm 341, 541	DKU 7140
Self diagnostics	Yes	Yes	Yes
Character sets	Arabic, English (U+L)	Arabic, English (U), (L) optional	Arabic, English (U+L)
Codes used	ASMO 449, ARCII	Comterm	ASMO 449,Comterm 341
Diacritics	Yes	Yes	Yes
Fonts (Arabic/Latin)	1 each	1 each	1 each
Arabic printing			
Modes: 1	Near letter quality	Near letter quality	Near letter quality
2	Draft	-	-
3	EDP	-	-
Matrix size (Latin/			
Arabic) for mode: 1	9x18 / 15x18	16x 8 / 16x18	/ 18x40
2	5x 9 / 7x 9	-	-
3	4x 9 / 5x 9	-	-
Printing attributes	Bold-underline	-	Double Strike
	Superscript-subscript	-	Underline
	Proportional		
Color printing	Yes	No	No
Graphics	Yes	No	Yes
Resolution	Medium: 72x 72	-	50 or 100 dpi
	High : 144x144		

printers are:
- o Microprocessor based, with buffering
- o One or more communications interfaces
- o Diacritics supported
- o Usually offered as hard-copy for terminals or as matching printers for arabized applications (mainly Word Processers)
- o Limited printing attributes
- o Generally only the quality level is available (LQ or NLQ).
- o Usually only one Arabic and one Latin font available for the same quality level.

(In one case the penalty for support of diacritics was the loss of the lower-case Latin character set).

The most important consideration is the necessity for at least two quality levels, draft and (near) letter quality. The handicaps of only one (usually high) quality level become obvious when long documents are repeatedly drafted and corrected before being printed in their final form.

5. REQUIREMENTS FOR ARABIC PERIPHERALS

Based on the evaluation results discussed in 4, a number of requirements for the design and development of Arabic (vs. Arabized) equipment can be identified. Before summarizing these requirements it should be emphasized that the most important single requirement in this area is to achieve standardization of the different Arabic and bilingual information interchange codes, of the ensemble(s) of Arabic character shapes, of different levels of C.A. algorithms and of hardware (e.g. keyboards) where possible.
In the following reference to standardized or to-be-standardized items is made as "ASMO" without specification of any particular ASMO standard. The following requirements are not all meant necessarily to be implemented in the same machine . They represent more the general frame including the essential as well as the desirable properties of each product. The choice of the properties to be included in any specific product is the task of the designer.

5.1 Arabic Microcomputers:

Table 8 summarizes the major requirements for an Arabic microcomputer (AMC).

The AMC should display Arabic characters and diacritics as a part of its native mode for display of alphanumeric characters, as opposed to the display of Arabic characters on a graphic screen.

It should have a minimum character resolution of 9x12 for text without diacritics and 20x24 for text with diacritics. A screen buffer is highly desirable, especially to support text editing.

The AMC should also allow full bilingual operation and possess Arabic graphics capabilities.

It might seem from the first part of this table, that the hardware requirements can be met readily. It should be pointed out therefore, that the main design challenge is

Table 8. Summary of AMC requirements

FUNCTION	Requirements	
	Home	Personal
Processors	8 bit	16 bit
Memory capacity	64 K	256 K
Cassette tape	E	X
Cartridge	I	X
Floppy diskettes	I	E
Hard disk	X	D
Display Matrix	9x14	20x24
Character shapes	G	VG
Contextual analysis	E	E
Display in color	I	D
Code used	ASMO	ASMO
Graphics	E	I
Resolution	Medium	High
Handling of numerals	E	E
OS Arabic commands	I	E
OS Arabic error messages	I	E
OS Commands with Arabic files	I	I
Arabized BASIC	E	D
Fully arabized programming language	I	D
Compiled BASIC	D	I
Arabized FORTRAN	X	X
Arabized PASCAL	D	D
Arabized LOGO	I	X
Applications: Spread Sheet	D	I
Integrated programming	X	E
DBMS	X	I
Access Arabic files	-	I
Access biling files	-	D
Graphic packages	X	I
Accounting S/W	X	I
Word processing	D	I
Communications	I	I
Education	E	D
Scientific S/W	D	I
Voice synthesis	D	D
Automatic translation	X	D
Price	Low	Medium
Manufacturer support	D	E
Compatibility new H/W, S/W	E	I
Physical portability	I	D

E = Essential
I = Important
D = Desirable
X = Not desirable

in implementing the required functions and simultaneously achieving the miniaturization and packaging density for the AMC to be portable enough, the reliability for it to perform in the generally rough environment (power, heat, dust, etc...) and the engineering to make it a medium to low cost product. The necessity of compatibility wit'in the same range of products and with rapid technological advances need not be stressed again.

On the software side, the AMC should preferably have a disk operating system that

can communicate with the user in Arabic or English, and that can access and process files with Arabic or English names. It should support at least one fully Arabic programming language. Because of the user category the AMC mainly aims at (home, education, personal, business), and because all microcomputers come with a BASIC language interpreter, either as an integrated part of its firmware (as in home computers) or disk based (as in personal and business computers), it should be a Basic-like language.

Since there is a fair population of Arabic programming professionals, and given the widespread use of languages such as PASCAL, FORTRAN and LISP, it is believed that the AMC should support versions of such languages that allow Arabic insertions. This would encourage the developments of more Arabic applications software.

Given the phenomenal success of LOGO as a computer language for children and beginners, it is believed that a Logo-type Arabic language is important. Finally it should be stressed, that there is a strong synergy between a given computer's popularity and the availability of a large prepackaged software library. This, in our opinion, will be the case with the AMC.

5.2 Arabic Terminals:

The hardware of the Arabic terminal (AT) should comply with the general requirements of the AMC hardware, including automatic selftest. The AT itself should provide buffering, display of 80 or 132 characters/line, large character display matrix (see 5.1) a large character generator for bilingual operation, and multiple video attributes. It should allow the firmware-based menu-driven configuration of the terminal parameters including the selection of an appropriate Arabic code set among several available, in order to permit the use of the AT with different Arabic software generations during the long transition phase towards standard code sets. A high-quality CA should be integrated in the AT.

Several (optional) communication interfaces in addition to the main serial (V24/RS232) interface should be supported. Asynchronous and synchronous versions should be available including master/slave terminals for polling operation and the major vendors protocols. An interface for a hardcopy printer should be provided. The capability of directing the host output to either the CRT, the printer or both is desirable. Programmable function keys should be provided.

5.3. Arabic Printers

Printing of Arabic or bilingual text would best be realized on standard off-the-shelf dot matrix printers with built-in CA and, either a standard Arabic character set, or capability for down-loading the appropriate set to the printer. The over-all control of the printing operation would be by the host, the AMC, or, in the case of hard copy, by the AT microprocessor.

At least two modes should be available : draft and quality printing as well as several fonts (Scripts) for the quality mode, if Arabic printing is to provide similar performance to that available on the same printer class in Latin mode (e.g. italic, gothic, elite, etc...).

Full printing attributes should be provided including enlarged and condensed printing. Graphic capability would help support a wider range of applications.

6. CONCLUSION

The present survey has outlined the major characteristics of the second generation of bilingual computer equipments, namely:

- o Hardware Arabization of equipment originally designed for Latin operation.
- o Software Arabization:
  - * Applications level mainly
  - * Programming languages: through insertion of Arabic text in input/output commands.
  - * Operating systems: first Arabic extensions to the BIOS of a few microcomputer OS, but commands, error messages and file access are still not Arabic.
- o Integration of propriatary algorithms for contextual analysis in the peripherals
- o Proliferation of Arabic code sets, slowing down with the publication of ASMO-449
- o Non-standard keyboard with reduced character set
- o Cost 30-40% more than equivalent latin equipment

Today we are on the verge of the third generation of bilingual equipment, as documented by the appearence of first fully Arabic microcomputers, programming languages, and operating systems, and by the growing movement of R & D on the subject both inside and outside the Arab World. This third generation will be characterized mainly by:

- o A much stronger penetration of standards, developed commonly by ASMO, equipment manufacturers and Arab scientific institutions.
- o Hardware specifically designed for Arabic and bilingual operation.
- o Software including:
  - * Fully Arabized operating systems.
  - * A few fully Arabic programming languages for widespread use.
  - * Main professional programming languages with Arabic insertion capability.
  - * A large library of applications including integrated packages for office automation.
- o More powerful contextual analysis algorithms integrating better editing, diacritics handling and spelling aids.
- o More graphics and color capability and related applications (CAD/CAM, CAE , CAI, etc...).
- o Character and word recognition applications for automated document entry.
- o Voice Analysis/Synthesis applications for input/output.

ACKNOWLEDGEMENT

The authors wish to express their gratitude to Dr. M. B. Mounajed for reading the proofs of this paper, and Miss M. Stephan for the extensive editing and formatting of the text.

REFERENCES

[1] Kotob, S., Arabic Microcomputers: An Assessement, Final Report ASD-7, KISR 1685, April 1985

[2] Harfouch, N., Project 207100AP09, Progress Report, (Arabic), SSRC, June 1985

[3] ECMA, Draft, Standard ECMA, Arabic Latin Keyboard Layout, ECMA/TC1/85/N29, April 1985

[4] ASMO, Draft ASMO Standard 663, Document 85-17 of the meeting of TC-8, Damascus, 14-16 May 1985

[5] Mackay, P. A., Observations on the Development of an Information Interchange Standard for Arabic and the Arabic Script Languages, PP. 4-5 , Tech. Report # 81-01-02, Dept. of Computer Science, Univ. of Washington, January 1981

[6] ARABURO, User's Manual, Arabic-Latin Technology, 1st Issue, March 1985

[7] Aton, Multilingual Arabic office management system, User's manual

[8] Mounajed M. B., Sandouk M. Z., Design Considerations of the OMRAN System, a CAD-system for Construction Engineering in Microcomputer Environment, Proc of the 4th European Conference on CAD/CAM and Computer Graphics, Paris, vol.2 pp.24-39, Hermes, Feb.1985

[9] Abou Al-Haija, A., and Idriss, M., Comparision between keyboard layout projects, (Arabic), pp. 1-10, Report, Univ. of Yarmouk, June 1984

[10] Mounajed, M. B. and Mrayati, M., Design Considerations of Arabic Peripherals, Proc of the Arab School of Science and Technology, 7th Summer session: Informatics and Applied Arabic Linguistics , Zabadani, July 1985

[11] Hyder, S. and Ibrahim, K., Optimal Placement of Isolated Characters on Arabic Keyboards, Proc. of the International Symposium for Standardization of Codes, Character Sets and Keyboards for the Arabic Language in Computers, Riyad, pp. 140-180, June 1980

[12] NCR DECISION MATE V, MS-DOS ARABIC User's Guide

Font Design for Computer-Based Documentation Systems

PIERRE A. MACKAY
Department of Near Eastern Languages
University of Washington
Seattle, Washington, USA

The old-style hot metal type-foundry is gradually vanishing from the world, except as a rather expensive craftsman's hobby, but the lessons learned from type-founding remain valid, and digital typeface production systems must in large part still work to emulate the punch-cutter's art. For the design of Latin-letter type fonts this emulation is unavoidable and even desirable. There is an immense wealth of historical experience behind even the most modern design, and almost all of that experience is based on the aesthetics of hot-metal typefaces. As soon as we move away from Latin-letter typefaces, however, we discover a great number of written character sets which never worked very well in hot metal. We have an opportunity to look again at these characters, and to adopt new approaches to them, free of the constraints that were inherent in physical characteristics of movable lead types. On the other hand, we face a different set of constraints, which require very careful evaluation as they affect type design.

In the early days of computer-assisted document preparation, the commonest systems for producing character output borrowed from the technology of the typewriter and involved the use of a raised letter face struck against the paper, which leaves a mark by driving ink out of a ribbon that passes between the typeface and the paper. The design of such typefaces was fairly close to traditional type design, but the way in which the ink was laid down made it impossible to get really fine distinctions. The exaggerated contrasts which are characteristic features of typefaces on the Bodoni model were tried out on some models of manual typewriter, but did not survive into the days of the "letter quality" printer.

An oddly analogous system was the one used by the early film-font phototypesetters, which derived their typefaces from copies of old hot-metal fonts. The solid old workhorse which produced reams of output from the Unix troff typesetting program used photographic negatives of type, and "flashed" a character onto positive film, instead of driving ink into paper. The typefaces were still the same in basic design, however, with smoothly curved outlines photographed directly off artwork, and transferred by a further photographic process onto the final positive film. The methods by which the film image was reproduced in ink on paper need not concern us here.

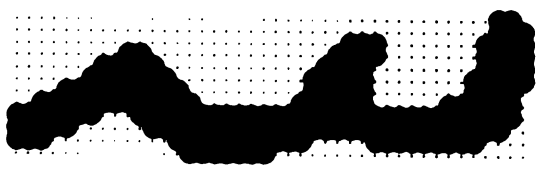
Both the character struck through an inked ribbon and the film image were transitional solutions to the problem of computer output of documents. They were supplemented, for a rather specialized group of devices, by characters formed from short vectors, suitable for reproduction on a plotter. The modern system, which is supplanting all other systems, is to produce each character by mapping it out as a region of dots of one color in a larger area of a contrasting color. This approach has the advantage that it can be generalized to all sorts of character output devices. Contrasting dots can be mapped onto a CRT screen, directly onto paper, onto a charged drum which will then pick up black particles to be passed onto a paper sheet, or onto phototypesetting film. The mapping may be more or less precise depending on the number of distinct regions that can be mapped into any given area, but the process is essentially the same. CRT screens tend, in North America at least, to be mapped at a resolution of 72 dots to the inch, on up to 200 or more. Printers which use inked ribbons or jets

of ink have resolutions in about the same range. Electrostatically charged drums began with a resolution of 200 to 240 dots to the inch, and have gone on to 300 and even 600 dots to the inch. Phototypesetting systems run from 975 at the low end, to a notional 5333 dots to the inch at the high end. At these resolutions, it is commoner to use close-set parallel lines traced on a CRT tube to project the desired image rather than discrete dots, but the underlying mapping is the same. Each part of a line traced by a phototypesetter CRT is used to expose some region of film which can be described as being roughly at the coordinates x and y on the paper.

The use of digital images for typefaces is a requirement of technology. It does nothing good for the quality of the typefaces, and often does a great deal of damage. Except at very high resolutions, where the eye can no longer detect the offset between one line of dots and the next, it is more difficult to get a satisfactory typeface design into digital form than into any other. Diagonals and curves are rendered crudely at best, and the designer must often depend on the mental correction that the viewer of a severely damaged image will make in an attempt to produce something like a familiar image.

Consider a characteristic letter, at one of the most common digital resolutions, 300 dots to the inch:

Pixel Width 42; Pixel Height 12.
Width 10.0000 pt.



Magnified in this fashion, the character is far uglier than it might be at a true size of 10 points, but the enlarged view indicates some of the difficulties of working with the industry quasi-standard of 300 dots/inch. Here we are assuming that an initial sigma can take the full EM width of a 10 point typeface, which works out at 42 pixels to the em. Subtleties of line are difficult at best, and it is nearly impossible to simulate the fine gradations of angle that occur in a well written Arabic character. When a 300 dot/inch font is used, the reader's eye actually sees something very close to what is illustrated here; it is only the mental interpretation that follows which can make an image like this the equivalent of a written Arabic letter.

At screen resolutions the problem is, of course far worse. No subtleties at all are possible, and the screen image bears only the most schematic relation to the original Arabic letter:

Pixel Width 10; Height 3
Width 10.0000 pt.

The character in this case is drawn in write-white pixels, since most CRT screens now use a system of dark characters on a light background. The essential units of meaning in a text can be conveyed by characters in either of these resolutions, but only at the price of eyestrain and general weariness. The interpretive functions of the reader's mind must work harder to make sense of badly distorted characters, and this is undoubtedly one of the chief reasons for the general feeling that the intensive and continuous use of CRT screens is physically debilitating.

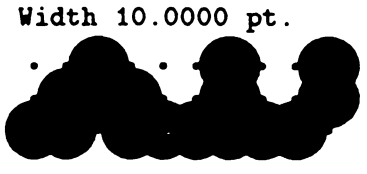
There are also problems involved in the sheer drudgery of creating characters for digital systems of reproduction. The normal industry procedure is to proceed from a specification of the outline contours of an ideal finished drawing to the description of a set of pixels which more or less approximates that drawing. At the lowest, most tedious level, the pixels may have to be counted individually. At screen resolutions, this is often the only satisfactory way to get a character-set tuned to make the best of the limited resources of the CRT technology. Moreover, although it may be a tedious task, it can be undertaken with confidence that it can be finished in a reasonable time. There are only 30 pixels to work with in the low-resolution character shown above, and a hundred of these would involve the discrete manipulation of only 3,000 pixels. In the printer-resolution character, at 300 dots/inch, there are 504 pixels, and a set of a hundred characters would therefore require the manipulation of 50,000 pixels. Under any normal circumstances this is not a reasonable task. (I once counted pixels for an entire Arabic Script font at 1800

lines to the inch, and I can recommend it as a cure for insomnia.) More efficient methods involve plotting curves that will fit as exactly as possible over the outlines of the ideal design drawing and leaving it to a program to fill in the pixels appropriately. In such systems the effort is focused on inner and outer contours, at the transitions between black and white areas of the character. Even when the contours are perfectly positioned however, the task is barely half finished. It is statistically unlikely that all contours will fall along the boundary between pairs of dots, and if they pass through the area of a dot, near the center, it is necessary to decide whether that dot shall be turned on (to contrast with the background color) or left off. That is not a mechanical decision that can be made solely on the basis of the precise area on either side of the contour. Except on gray-scale devices, such as certain CRT displays, a dot must be either on or off, but the blackening of any one dot can lead to an apparent intensification of the blackness of its neighbor, and cause unexpected visual effects. Consider the character below, which differs from the previous image only by the removal of the dot on the lower left corner. The attempt to simulate the curve under the first stroke is a disaster, and breaks the character up into separate parts.

Pixel Width 10; Height 3
Width 10.0000 pt.

On the other hand, if we were dealing with a display technology that emphasized the formative dots of the letter (write-black digitization) the image, while still crude, would at least be continuous.

Pixel Width 10; Height 3
Width 10.0000 pt.



The systems which derive digitized images from outlines are in some sense a computerized automation and extension of the traditional punch-cutter's methods Like the punch-cutter, these programs do not themselves generate the art-work; they transfer the art-work from one medium to another. There is often a good deal of handwork involved in establishing a close fit of the outline curve against the original art work. Splines and segments of spirals are both used by various commercial typesetting firms and font-suppliers, who will supply compact, and usually encrypted versions of outlines together with the software that converts these outlines into the digitized form that can drive the final typesetting device.

The firm URW of Hamburg, Germany, with its IKARUS system provides a good example of the use of a system of fitting outlines to existing artwork. Bitstream, of Cambridge, Massachusetts offers another, and also provides finely tuned bit-mapped fonts for many devices. AM/Varityper and Alphatype, both manufacturers of phototypesetting equipment use systems which are tied to their specific proprietary typesetter architectures.

These systems are often capable of manipulating the outlines before the final process of digitization. They can stretch, compress, and rotate both inner and outer contours to generate variations of an initial design, but they do not prepare the initial design.

An more radical approach to the production of digitized fonts is represented by PostScript, developed by Adobe Systems of Palo Alto, California. PostScript treats font generation as a part of a more general graphics language, but a highly specialized part. It is possible for users of the PostScript graphics language to create characters by using the primitives of the language and recording them in a form that will be recognised by other parts of PostScript as a character font. It is clear however, that although such user-generated fonts share many characteristics with Adobe proprietary fonts, in that they can be rotated, scaled and otherwise manipulated, they are not entirely the same as the proprietary fonts, which appear to have special parameters to govern subtle changes in the font when it is scaled.

Finally there is the METAFONT system, developed by Donald Knuth as a complementary part of the TEX typesetting system. Like TEX, METAFONT has been put into the public domain, and it is therefore possible to

illustrate its use quite freely. Like PostScript, METAFONT is a design tool in its own right, rather than a system for capturing the details of externally produced artwork. METAFONT character descriptions differ from most outline descriptions of font characters in that the normal point of reference for a METAFONT character is along the mid-line of an imaginary pen-stroke. The locations of the black-to-white transitions on either side of the midline are, of course, critical for the designer, and the current version of METAFONT gives much greater attention to that question than its predecessor did, but the fundamental metaphor of the pen-stroke is still deeply embedded in all aspects of the program.

It seems at first glance entirely natural to describe a character as if it were produced by the motion of a pen, and the concept has already proved itself in use. It is powerful enough to provide an experienced designer with a very satisfactory repertory of graphic shapes from which to build both text and display fonts. The "virtual pen" of METAFONT can be altered in shape; it can be tilted, made wider or narrower, or even re-cut to produce some entirely arbitrary contact surface at the nib. If the principles and methods of typeface design were identical with those of calligraphy, the concept of the "virtual pen" would in itself be sufficient. For some fonts it appears to be a very good concept but for others it is not.

As it happens, the most obvious case of a font for which the "virtual pen" concept is not in itself sufficient is the one for which METAFONT was initially intended. The vast preponderance of English language reading matter is printed in some variety of serifed Latin-letter typeface. Not one of the commonly used varieties of these faces, however, is the direct product of a calligraphic design. Every face produced before the 1950s was brought into final shape by some process of engraving, and many of the most sophisticated new techniques tend to emulate the action of the graving tool rather than the action of the pen, and the current version of METAFONT provides extensive pacilities for controlling the outline contours of a character, where this is most appropriate.

The path of a METAFONT pen-stroke is described abstractly, through the use of Cartesian coordinates, and it is often something of a surprise to the designer to see what METAFONT has come up with on the first attempt. It is very difficult to get an exact match with an artist's drawing, and takes a number of corrections in the original path specifications. For this reason, METAFONT will never be a good system for copying existing fonts. Moreover, the most successful work with Latin-letter characters will tend to look more like the results of engraving than of calligraphy. Inscriptional models played a great role in the development of Latin-letter textfonts. A wealth of stone-cut inscriptions from the Roman Imperial centuries was available throughout Western Europe during the formative period of movable type design. The importance of such models can be seen in the contrast between the rich and satisfactory variety of good Latin-letter text fonts and the rather meagre repertory available for Greek. For several centuries, the designers who worked on Greek had little or no direct experience of stone-cut Greek characters, and they continued to depend primarily on manuscript styles until well into the last century. The traditional, but quite unnecessary distinction between medial and final sigma in Greek alphabets is a last reminiscence of the long adherence to manuscript ligatures in Greek text fonts. (There is, incidentally, no single-word translation for the terms "font" or "typefont" in Greek.)

As we get even further distant from the Latin-letter alphabet, we come to languages whose graphic character set is almost exclusively calligraphic, without any significant contribution from stone-cutting techniques. Many of these non-Latin character sets were severely constrained by the requirements of design for movable type. Some elegant compromises were made but they were always compromises. Perhaps the greatest single example (leaving out the very different Chinese ideographic character set) is the Nirnaya Sagar font of Devanagari characters for Sanskrit. A really full set of Devanagari in a single style could run to over 1000 separate sorts, and even then an exacting critic might well notice the omission of certain combinations of conjunct and vowel. The demands of modern printing have seen a

steady decline in the number of characters used, and I do not know of anyone who has argued that this decline improves the appearance of typeset Sanskrit texts.

In a study published at Stanford University, Mr. Pijush K. Ghosh, of the Tata Institute in Bombay, explored the use of METAFONT for a new Nagari font. Ghosh was working with a character set that is basically an expression of penmanship, and his approach was noticeably different from that of a Latin-letter designer. He used a very limited range of pen-widths, and achieved a variety of styles by varying the shapes globally. The curves in his initial design are mathematically simpler than those used for the archetypal Latin-letter metafont, Computer Modern, but they appear to a non-native eye to give a very pleasing result. With the additional design contribution of Mr. R. K. Joshi, it is easy to see that an important new Nagari typefont can be developed from this beginning.

After several false starts, work at the University of Washington, in Seattle, Washington, is proceeding on a METAFONT character set for the Tamil Language. The smooth flowing curves of Tamil are even more natural to the METAFONT system than the curves of Devanagari. Tamil forms were developed for ease of writing with a blunt stylus on palm-leaves. Sharp angles and tight curves had to be kept to a minimum, and straight lines had to avoid the weakest direction along the leaf fiber. The result is this extremely good-natured effect. There are tremendous possibilities here for a designer, particularly in the relation between the initial closed loop which is a feature of about half the consonantal alphabet and the remaining curves. In these examples, we have used the simple circular pen, which more or less keeps the feeling of the historic stylus. Changes in pen shape will introduce the possibility of differentiating vertical from horizontal stroke weights, and that appears to us to be the first necessity for refining this into a good serviceable text font.

The work on the two Indian character sets (Devanagari for Hindi and Sanskrit, and Tamil for the South Indian scripts) is a clear example of the benefit of using a design system based on the simulation of pen-strokes when it is applied to a character set based on handwriting. An even more effective example is the case of Arabic Script.

All recognized forms of Arabic Script are based on penmanship, and even the finest efforts in hot metal or hot-metal emulation have always been constrained by the difficulty of imitating penmanship in a system based on the traditions of movable type. The current version of METAFONT may not eliminate all these difficulties, but it is clearly a great step in the right direction.

In Arabic Script we may state it as a general rule that the graphic shape of every consonant is potentially affected by the shapes of the preceding and following consonants. This variability reflects the history of the script. While most of the surviving writing systems of the Mediterranean world developed block character sets suitable for being incised or pressed into the writing medium, Arabic developed directly as an efficient penman's script, and all the many accepted forms of Arabic writing are descended from pen-and-ink originals. Over several centuries, certain specific adaptations have been developed to provide for efficiency of production and increased legibility, and these are common to all the styles of writing in general use today. Even a superficial glance at passages in *Naskhi*, *Ta'liq* and *Ruq'a* will show as much kinship as difference, and the common elements may be taken as a good practical indication of the basic outlines of context-sensitivity in all Arabic script texts. The readers of these scripts have in the past been offered any number of plans for radical "simplification" of the script, but they have always rejected them. It is not only that the shapes of Arabic are sanctified by tenets of religious belief, though that is one of the most important considerations, it is also that some features of context-sensitivity contribute substantially to the readibility of an Arabic script text. The context-sensitivity of a writing system such as Arabic is an entirely different matter from the minor variants and ligatures which can be found in most other scripts.

The various manifestations of context-sensitivity in Arabic script may be classed in the following rough order of importance, giving particular weight to considerations

of increased readability.

1) Differentiation of initial/medial from independent/final forms.

2) Provision of the *lam/alif* composite character.

3) Provision of all combinations of the character *hamz*^{ah} with the appropriate "supporting" character.

4) Conversion of the sub-alphabetic sign of feminine gender in Arabic nouns and adjectives, *hā' al-ta'nīt* (also known as *tā' marbuṭ*^{ah}), to the forms of alphabetic *tā'* (*tā' ṭawīl*^{ah}) when it appears in non-final position.

5) Provision of superscript (rather than adscript) vowellings (*ḍamm*^{ah}, *fatḥ*^{ah}, *kesr*^{ah}, their "nunated" or doubled forms, and *sukūn*), along with the sign for the doubled consonant (*ṣadd*^{ah}).

6) Differentiation of initial from medial and of independent from final forms.

7) Provision of some vertically stacked consonantal ligatures.

8) Provision of certain long consonantal forms to fill out a loose line in left-justified text. This is by far the most aesthetically satisfying method of justification, but it has scarcely been considered by any computer assisted typesetter/formatter up till now. All previous systems including KATIB have relied on:—

9) Provision of extensions of the join-line between certain Arabic script characters (often referred to by the Persian term *kaşīde*, meaning "stretched, extended"). This produces a most unsatisfactory effect, since it introduces a hard horizontal rule into an otherwise gently curvilinear script. It violates all the natural aesthetics of any form of Arabic Script except for some of the recently designed forms of so-called "neo-Kufic." We will make little or no use of this technique in the Arabic Script version of TEX.

A font that reflects all these considerations is going to be a very large one. Ten years ago, when we produced the **KATIB** typesetter system at the University of Washington, we found a need for 24 different shapes of the letter *bā'*, 27 shapes of *nūn* and a similar number for *yā'*. We actually did digitize those shapes by eye, with reasonably satisfactory results, but we could not face the work of producing more than one typesize or style. Even the effort of refining an unsatisfactory character became an unbearable prospect, and there are several excessively dense characters in the one book published with the **KATIB** font which we were well aware of at the time. Nor could we adapt the one effort of digitization to any other resolution. Mechanical schemes for pixel-shedding are probably the worst of all possible ways to reduce the resolution of an already digitized font unless you have some very sophisticated software to correct anomalies and enforce consistency of stroke-weights across a variety of characters Even METAFONT, working from an abstract description, has its weaknesses at the lowest and most difficult resolutions, and at resolutions lower than 100 dots/inch, it needs the help of a pixel editing program to produce a satisfactory result.

The results of using METAFONT for high-resolution devices, however are entirely satisfactory, and it is possible to exhibit here some of the preliminaries for a new Arabic font. Since we are using the Calligraphic model, we need first to cut a pen. Fig. 1. shows the pen as seen from inside the inkwell (without any perspective correction).

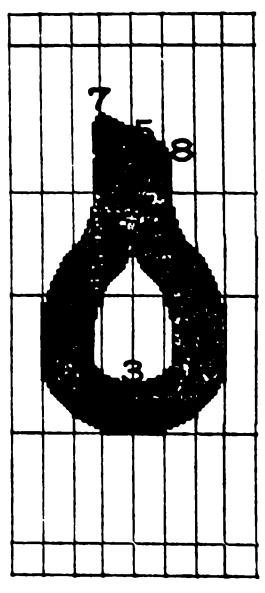


Figure 1. Cut end of reed pen

After dipping this into the ink, we touch it to the paper, and get the following mark, which is basic to the entire font.

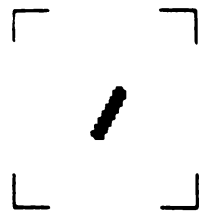


Figure 2. Nib contact surface

Drawing it down at a 45° angle produces the *nuqṭ^ah*, which provides a basic scaling element for the entire font. Six of these measure the internal length of the final *bā'*.

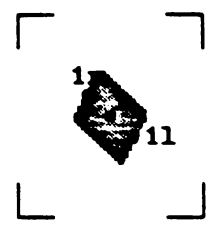


Figure 3. dot

At this point we can establish several critical dimensions for the font. We adjust the size of the *nuqṭ^ah* so that eleven *niqāṭ*, loosely stacked, fill the body of a reference grid representing the typeface (leaving a small border for overshoot at both top and bottom). The baseline is set to allow *alif* the height of seven *niqāṭ* and to allow a depth of four. We draw a row of six dots along the baseline to establish the width of *bā'*.

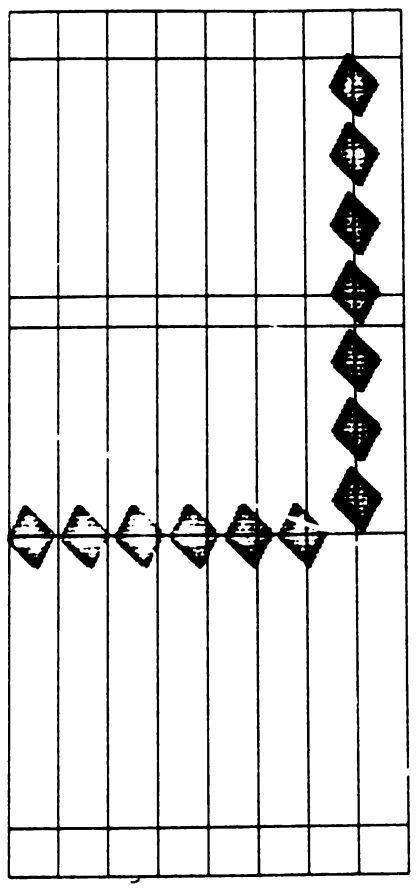


Figure 4. Basic proportions

Finally we can draw two sample characters,

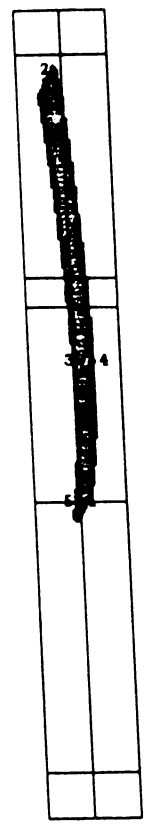


Figure 5. The letter alif

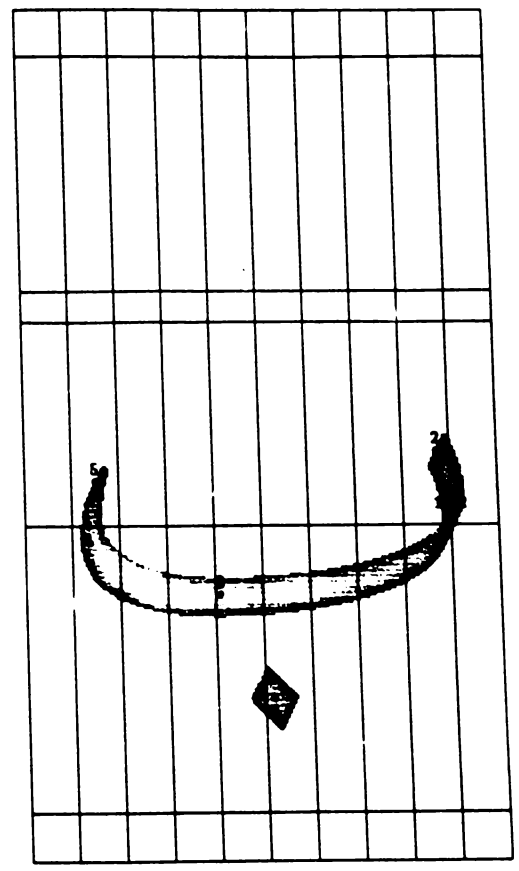


Figure 6. the letter bā'

From now on, all sizes are stated in terms of the dimensions of the *nuqṭ^ah*, in the best traditional manner. There are no other letters coded as yet in the current version of METAFONT, so the following examples have had to be taken from the old program. The

syntax of old METAFONT is quite different and the new program is very much more powerful, but the general principle remains the same.

Rā' is a very simple character in construction, although it turns out to be one

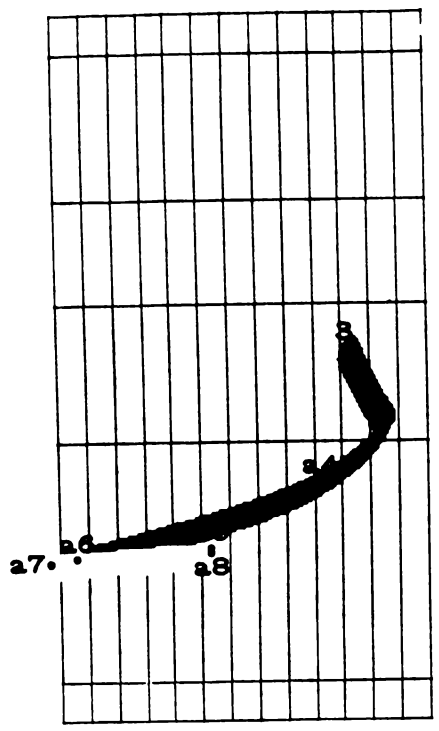


Figure 7. The letter rā'

of the most difficult in design. (*Dāl* is even more subtly frustrating.) The upper part of the stroke is drawn through three points, of which 3 is clearly visible on this illustration, while 2 and 1 are half hidden by the stroke itself. The direction at each point is slightly forced, to give a tension to the line, and at 1, where the tail begins, the pen is actually being drawn vertically downward. This gives a slight "heel" to the stroke which eliminates an otherwise bland transition to the long curve into the tail. The tail is really quite elaborate. There is a control point visible just above a8 on the illustration. The pen does not pass through this point, but it aims at it just after it leaves the "heel" above. That pulls the line down into a more pleasing curve than a direct path would give. The treatment of such control points in the new METAFONT is even more sophisticated than that available here, and it is also possible to change the "tension" in all or part of a curved path. Much of the improvement in this feature of METAFONT results from the work of John Hobby on the development of a set of subroutines for Chinese character strokes.

At the tip of the tail, a calligrapher would roll the pen and lighten the stroke to get the long taper. That is computationally too expensive, and the use of an "eraser" produces the same effect much more efficiently.

Many common elements of characters can be saved for future use. In the old METAFONT the mechanism was subroutine calls. The new METAFONT partly to retain a close similarity with TEX, handles all such common elements through macro substitution.

In the letter *mīm* only the loop had to be designed. For this variant of independent *mīm* the tail of *rā'* can be added as a common element.

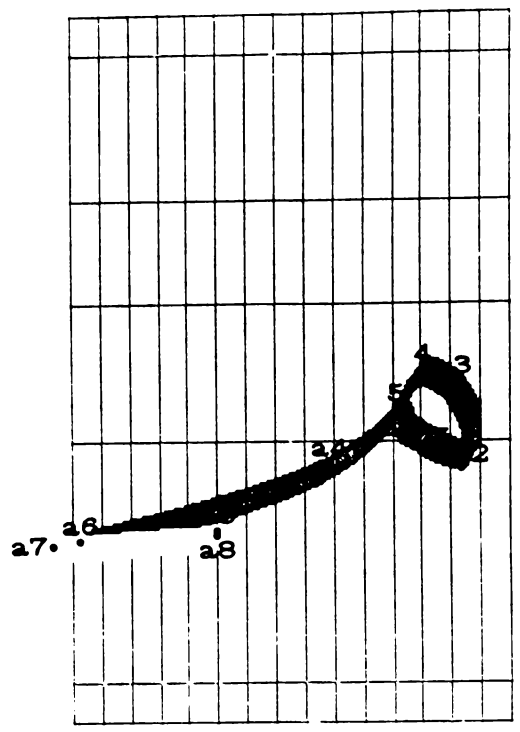


Figure 8. the letter mīm

The *ṭā'* provides a good example of the standard vertical stroke, which when lengthened slightly provides an *alif*. You may just be able to see a second dot to the right of a3 on this character. That is another control point. By moving this left and right, it is possible to govern the inflection of the stroke in very precise ways. Point a13 is one end of the eraser used to taper the lower part of the stem. If you look carefully at the character in the caption for figure 9, you will see that it does not taper. This is the sort of compromise that is necessary when you take a high-resolution character and reproduce at 240 dots to the inch. I discovered that the bottom end of these stems was attenuated to a single pixel width, and it was clear that that effect was considerably less desirable than a

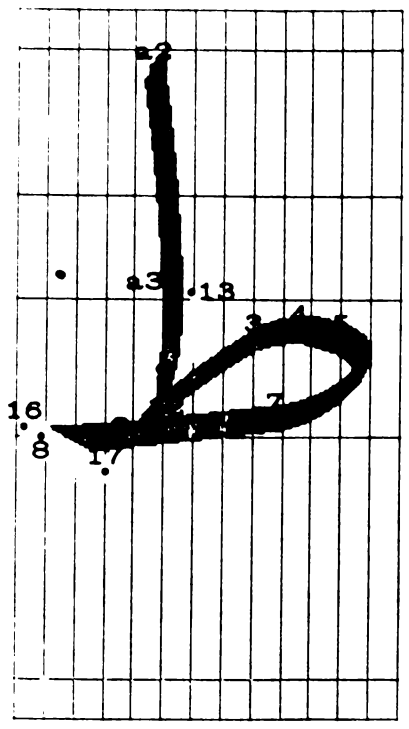


Figure 9. The letter ṭā′

coarsening of the line.

In the next three characters, we see another use of a common element in several different letters. In theoretical treatises

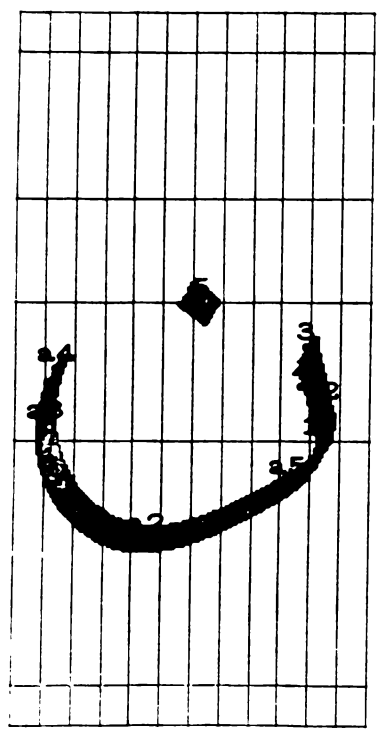


Figure 10. The letter nūn

on classical calligraphy it is stated that the loop of *nūn*, *qāf* and *lām* are exactly the same. This rule may have been followed in some very formal hands such as the one used as a model for the **KATIB** typeface in 1975, but it produces a cold and rather hard looking effect, with a *lām* that appears too wide and, when the same recipe is applied to *yā′* a *yā′* that looks too narrow. When we have the resource of parameterized subroutines or macros, it is possible to vary the dimensions of the strokes in question and produce what the calligrapher usually produced—a variant on an essentially similar stroke. It these three examples, the large loop of *nūn* and *qāf* is given the same proportions, but the loop of *lām* is shrunk slightly.

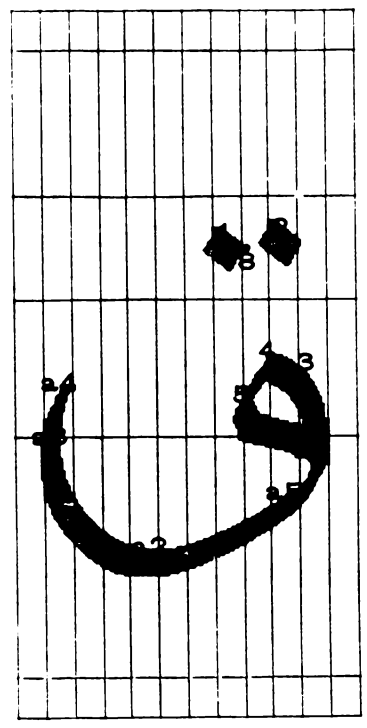


Figure 11. The letter qāf

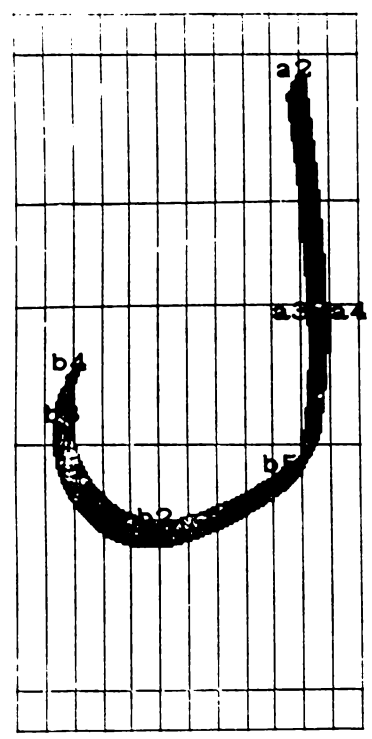


Figure 12. The letter lām

References

Adobe Systems Incorporated. *PostScript Language Reference Manual*. (Addison Wesley, Reading, Mass. 1986)

Knuth, Donald. *Computers and Typesetting*. Volumes A-E. (Addison Wesley, Reading, Mass. 1986).

Design Criteria of Arabic Peripherals

M. MRAYATI and B. MOUNAJED
Scientific Studies and Research Center
P.O. Box 4470
Damascus, Syria

Abstract

This article presents the main problems involved in the use of the Arabic language in computer peripherals. Arabic character repertoire, diacritic and codes are discussed. Basic criteria which are to be taken in considering the design of Arabic peripherals are proposed. Some examples on the bilingual keyboard layout, Arabic and bilingual printer chain, and Arabic daisy wheel are given.

1 - Introduction

The Arabic alphabet is the second most widely used alphabetic writing system in the world. Up till the 19 th century the Arabic writing system was the system most studied. Several writing styles were defined and used extensivly. Each style has its own well defined rules. When the print process appeared, Arabic typography adopted certain machine compatible fonds , several simplification steps took place. The Naskh style is commonly used by the Arabic printing industry because of its simple ligature rules.

When computer print out first appeared, Arabic script was not introduced adequately. Computer use is spreading at a much slower rate in the Arab world than in western or far-eastern societies. Furthermore, the Arab scolars or scientific institutions are not fully involved in the "computer revolution". They are not offering designs or solutions necessary to the correct use of the Arabic language in computer hardware and software. The result is the introduction of the Arabic language in computers in a dubbed manner and in an environment designed for English.

In this study, we concentrate on the Arabic input-output requirements. The main criteria necessary to assure optimal design of I/O peripherals are stated. Some examples are given of the application of these criteria. We would like to emphasise that the final solution,which is user acceptable,is that which gives good quality computer output and not any other solution that sacrifics the basic aspects of Arabic script.

2 - The Particularities of Arabic Script

Contrary to the English script Arabic has the following particularities:

a - Arabic script is entered and displayed from right to left.

b - Arabic letters have more complex caligraphy.

c - Arabic letters are always connected.

d - In addition to the basic set of letters, Arabic script uses diacritics, which require a larger character matrix size

e - The character set has to be dealt with at different levels. One is led to have a basic set at the keyboard and memory levels, and another set at the output level. The second set is an extended set including the different shapes of the letters of the basic set depending on the letter position and the printing style.

f - Arabic text has no hyphonation.

g - Numerals are written from left to right, namely, opposed to text writting direction. Depending on the country, numerals have two shapes Indian or Arabic.

h - Arabic characters show pronounced variations in width. Some characters may be several times the width of others such as lam and alef.

Consequently, the Arabic input-output peripherals should satisfy the following basic requirements:

1 - *Entering and displaying Arabic script from right to left.*
2 - *Large character matrix for displays and matrix printers (at least 10 x 16).*
3 - *Inclusion of complete diacritics.*
4 - *Bilingualism (e.g. Arabic + English).*
5 - *Contextual analysis algorithm including numeral analysis.*
6 - *Taking into consideration the frequency of occurance of Arabic characters depending on the application.*
7 - *Standards should be established and followed in order to achieve compatibility and transport - ability.*

Some of the essential standards are:

a) *7-bit code ASMO 449 (though the control character for changing modes and for inserting Arabic into Latin and Latin into Arabic are not yet defined).*
b) *8-bit code for data communication and for data processing.*
c) *Arabic repertoire (which is proposed by this study to ASMO).*
d) *Contextual analysis algorithm which deals with numeral handling in a standard way (if b and c are standarized, we need only to define some of the aspects of this algorithm).*

3 - The Arabic Character Repertoire

In order to design and standarize the use of Arabic characters in computer I/O, one should consider the I/O process as a system. Basic system theory would aid in designing and standarizing this process. The input to the computer (through the keyboard or via a transmission media) should follow a fixed standard, such as ASMO 449 in 7-bit bilingual invironment. The output of the system is the total Arabic character repertoire and the contextual analysis algorithm realizes the transformation between the input and the output as shown in figure (1).

The three parts of the system are interdependent; definition of two of them will determine the third . If we assume that the input is coded using the ASMO449 standard, then definition of the output Arabic character repertoire will determine the contextual analysis algorithm, and consequently the whole system could be standarized.

The Arabic character repertoire at the output will depend on the printing style used. There are several well known styles and the main ones are: Naskh, Thouloth, Roukai, Dewani, Koufi and Farsi.

In this paragraph, we will try to arrive at a standard repertoire which has a number of characters that can accomodate any of the wanted styles. The procedure or rules followed to reach this standard number are the following:

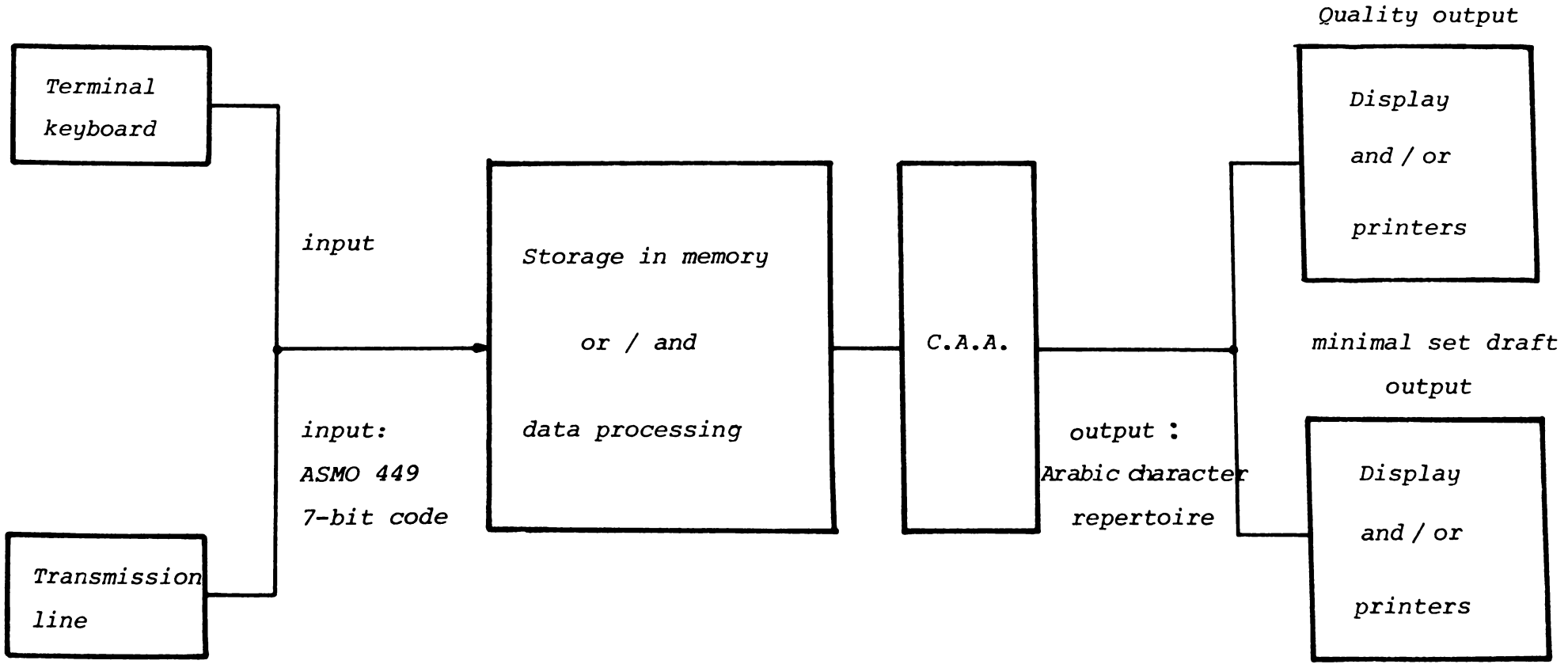


Figure (1) . The relation between the input set of characters, (example: ASMO 449), the Contextual Analysis Algorithm (C.A.A.), and the output Arabic character repertoire standard (proposed to ASMO).

1 - Find the style in which ligature is achieved without change of line level and in which the shape of character depends on the position of this character in the word (initial, medial final connected, final isolated) and not on the preceding or following character.

2 - Find the minimum number of character shapes in each style by applying certain rules while keeping the originality of the style.

3 - Take the maximum common set of the numbers found in (2) above and this will be the standard number of the Arabic character repertoire capable of incorporating any style.

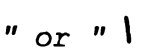
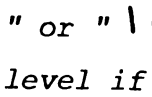
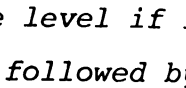
4 - The diacritic signs adopted are the same for all styles and their number depends on the input standard used and the contextual analysis used.

Firstly, applying rule (1) on the main Arabic printing styles will eliminate,(at the present stage of technology used in output peripherals and for practical and economical reasons), The Roukai , Dewani and Farsi styles. As an example the Roukai style is a very artistic style, but it has very large number of character shapes. The shape of the character changes depending on its position in the word and on preceding and following characters. For example characters such as "ب ت ث" have five shapes at the initial position of a word:

and five other shapes at the medial position:

in addition to two shapes at the final position:

 connected  isolated

Moreover, in the Roukai style the ligature is complex: The vertical positioning of each character chages depending on the preceding and following character. For example the character ـهـ is written on the line level if followed by "د" or "ا"  but is written higher than the line level if followed by "ر"  , and even higher if followed by "ح"  . Therefor, applying rule one will make us retain the three styles: Naskh, Koufi and Thouloth.

Secondly, applying rule (2) on the three chosen styles,we arrive, using the diagonal segmentation of characters and other simplification procedures, at the following minimum number for each of them:

Style	Minimum number of characters
Koufi	69
Naskh	85
Thouloth	86

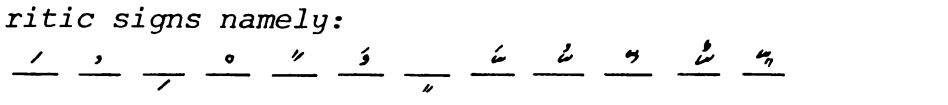
Tables 1, 2 and 3 show the shapes of characters corresponding to the three styles in their minimum number.

Thirdly, from rule (3), it is evident that the common set is 90 shapes which is proposed to be the standard number of Arabic character shapes. With this set, it will be possible to standardize the following:

Character generator(hard or soft); printing chains ; daisy wheels, and dot matrices.

In addition, any one of the three styles can be used without changing the standard.

Fourthly, the number of diacritic signs is the same for the three chosen styles. There are 12 diacritic signs namely:



We will discuss the problems involved with these signs in the next paragraph.

Finally, the standard special symbols, used in computer printed output, in addition to the numerals, form a set of 42 symbols as shown in table (4). Consequently, the complete repertoire of symboles is 90 + 12 + 42 = 144. If the Latin alphabet is also added for binigualism, the repertoire will contain170 or 196 symbols.

4 - Diacritics

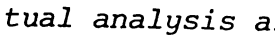
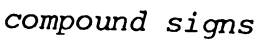
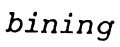
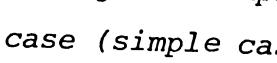
Diacritics should be looked upon at three stages: the input level (keyboard), the storage level(memory and processing) and the output level (output of contextual analysis algorithm). At the input level, the ASMO 449 which is a 7-bit code includes the following eight simple diacritic signs:  . As we have already seen, Arabic has five other compound signs which are  . At the output, the contextual analysis algorithm can generate these five signs either directly, if they are included in the standard repertoire, or by combining two simple signs if they are not. In the first case (simple case), the sign is printed in one step while we need two steps for the second case (composite case), namely one step to print the  and second step to print one of the following signs  . Notice that the last two signs are different from the corresponding simple diacritic signs

Table (1): The Thouloth style. Possible minimum set is 86 shapes. We add 4 shapes to have common standard repertoire of 90 shapes.

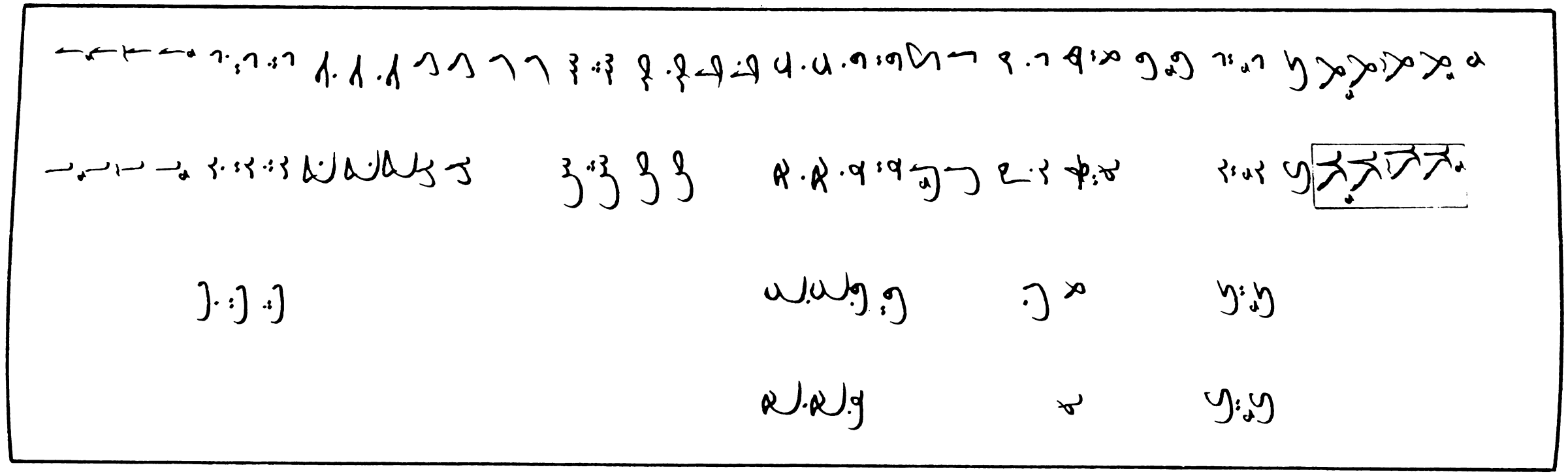


Table (2): The Naskhi style. Possible minimum set is 85 shapes. We add 5 shapes to have the common standard repertoire of90 shapes.

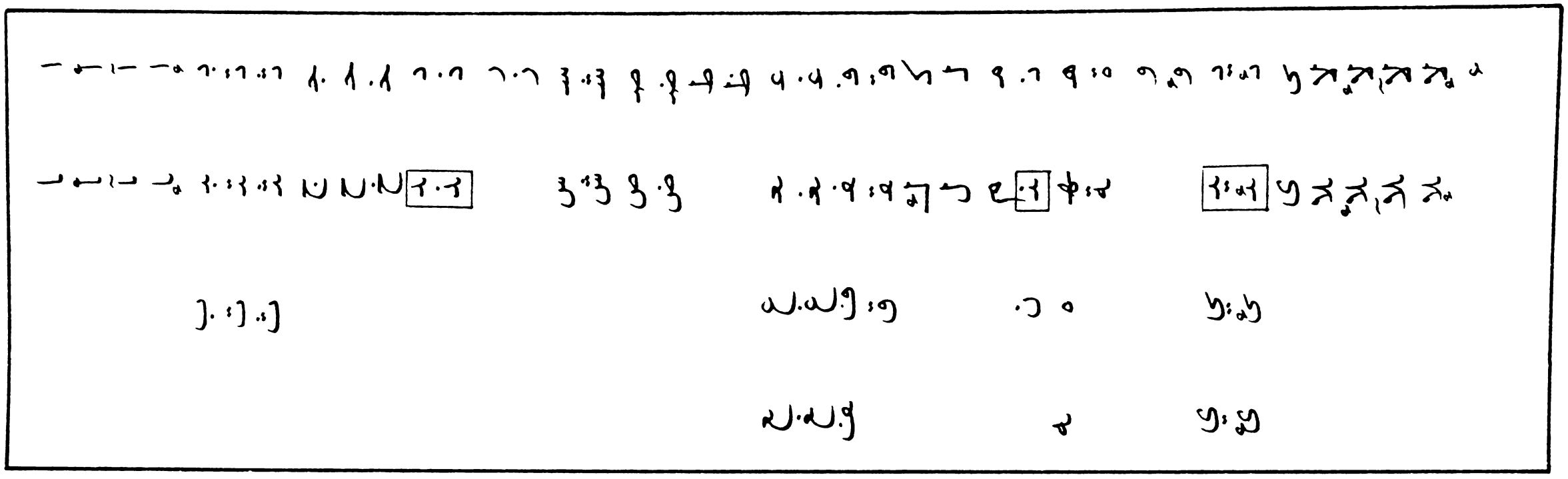


Table (3): The Koufi style. Possible minimum set is 69, namely the smallest of all styles. We add 21 shapes to have the common standatd repertoire of 90 shapes.

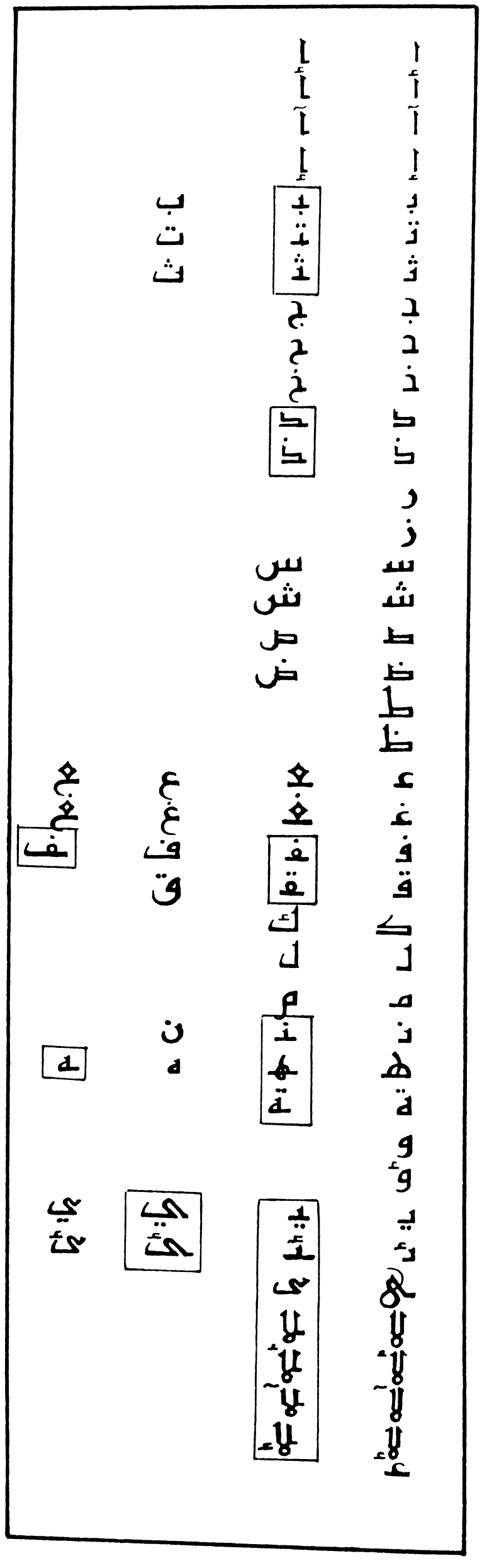
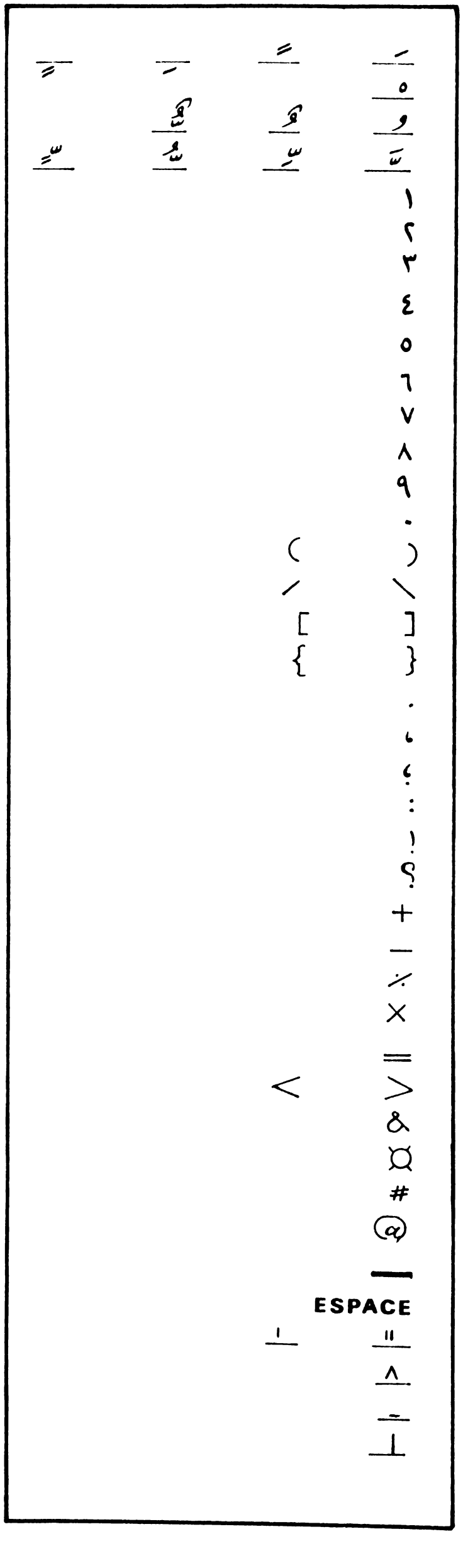


Table (4): The diacritic signs of the repertoire are 12, the numerals 10 and the special characters 32, namely a total of 54.



ـٍ ـٌ ـّ , and in this case they should be added to the repertoire. Notice that the compound sign ـّ does not exsite as such, since it is written as in the following examples: دسّاً ، مدّاً
Another problem with computer printing of diacritics is their placement relative to their corresponding characters. Traditionally, diacritic signs are always printed over and under the concerned character. We call this the vertical placement of diacritics . Recently, several proposals where made to put the diacritic signs after the concerned character. We call this the horizontal placement. This placement means that each diacritic sign is assigned one character position on the paper or on the screen. There are two reasons behind this proposal. First the relatively small vertical dimension and resolution of the printing space of a Latin character. Second, the need, in certain applications to have exact concordance between the number of symbols entered at the keyboard, stored in memory and displayed or printed at the output. When we use vertical placement of diacritics we do not have the same number of characters in memory and on the printed line. Table (5) shows examples on the two types of placement. It is obvious that we have, for the two types, a difference in representation between memory and printout.

Table (5)

	In memory	On the screen or paper
H.P	دَسَّ دَسَّا	دَسَّ دَسَّاً (11 places)
V.P.	(13 places)	دسَّ دَسَّاً (6 places)

Table (6) summarizes the advantages and disadvantages of each of the above mentioned cases. When this table (6) is studied taking into account the system point of view of figure (1), one can deduce as an optimal solution the following:

a) The addition of the compound diacritics to 449 which solves the problem of difference in representation between memory and printout.

b) The adoption of the vertical placement of diacritics which makes Arabic display or printout normal and improves user acceptance.

c) When diacritics are fully included in Arabic text it is worth considering the storage of each character with its corresponding diacritic as two bytes. This approach gives the ability to modify or delete diacritic signs without affecting their corresponding characters. Furthermore, the number of characters per line on the screen or printout is kept constant whether they are with or without diacritics.

5 - Arabic Daisy Wheels Printer

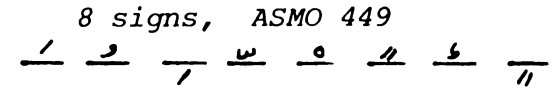
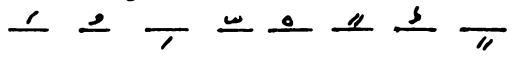
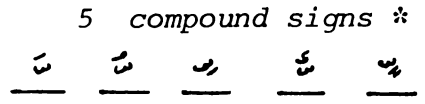
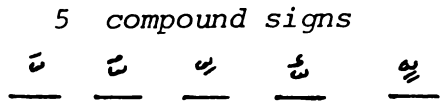
Daisy wheel printers are ideal for high quality output and are widely used in office automation environment. One good example on the usefulness of standardizing the Arabic character repertoire is the possibility to standardize an Arabic daisy wheel. One can change the writting style from Koufi to Naskhi or to Thouloth by simply changing the wheel of the printer. In this paragraph we will propose an optimal repartition of the Arabic character repertoire on the daisy wheel. This repartition is the same for all three mentioned styles. It is optimal from the point of view of speed, since it takes into account the frequency of occurrence of each Arabic character in the initial, medial and final positions (connected or isolated) it also takes into account the frequency of occurrence of bi- characters.

The optimization program is written in Pascal where the input is the character repertoire (102 characters) and the statistical results on occurrences of each character and the frequency of occurrence of the combinations two characters taking into account the position of characters (initial + medial, medial + medial, medial + final connected, medial + final isolated). The program also supposes the following specifications concerning the wheel movement:

- When a character is printed, the wheel stays at the same position.
- The wheel turns clock-wise and counter clock-wise and has two levels of characters.
- The wheel turns a maximum of 180° in each direction. The firmware of the printer compares the present position of the wheel with the position of the next character to be printed and consequently determines the direction of turning, the number of steps and the level of the wheel.

The algorithm determines the placement of all characters on the wheel by minimizing the distance between characters having high propability of occurring together. The statistics characterizing Arabic character occurrences were deduced from a study using texts totaling 200,000 characters. The samples forming

*Table (6) : Possible solutions for incorporating diacritics. The cases marked by the star * is our best proposed solution.*

Characteristics \ Solution		Simple solution	Composite solution
Assumed number of diacritic signs in code table		8 signs, ASMO 449 	8 signs, ASMO 440 
Number of signs to be add at the character generator level		5 compound signs * 	2 simple signs 
Number of signs of the character repertoire at the printed output		5 compound signs * 	5 compound signs 
Number of printing steps for the compound signs	Horizontal placement	One or two steps depending on the position of the sign if it is with or without (-) wasla	Tow or three steps depending on the position of the sign if it is with or without (-)wasla
	Vertical placement	One step always *	Two steps always
Difference in representation between memoery and printout	Horizontal placement	No difference for simple signs and one space difference for each compound sign	No difference for simple signs and one space difference for each compound sign
	Vertical placement	One for each simple sign * and two for each compound sign	One for each simple sign and two for each compound sign

the analysed text were taken from different sources (journals, books, newspapers,...etc.) [ ]. The text was with diacritics. We weighted the diacritics by a factor of 30% which is estimated to by the percentage of use of diacritics in different applications. This means that we assume the percentage of applications which uses diacritics to be about 30% of all applications. Or, that diacritics is used over 30% of the characters of a text. Numerals and special symbols are placed in one region of the wheel, with no special rule , since their use is highly dependent on the application. Figures (2) (3) and (4) show three possible standard daisy wheels including the proposed Arabic character repertoire. The 90 characters of the repertoire are distributed on the wheel according to the result found by the above mentioned optimiaing algorithm, taking into account the 12 diacritics.

Figure (2): Standard daisy wheel using the Thouloth style.

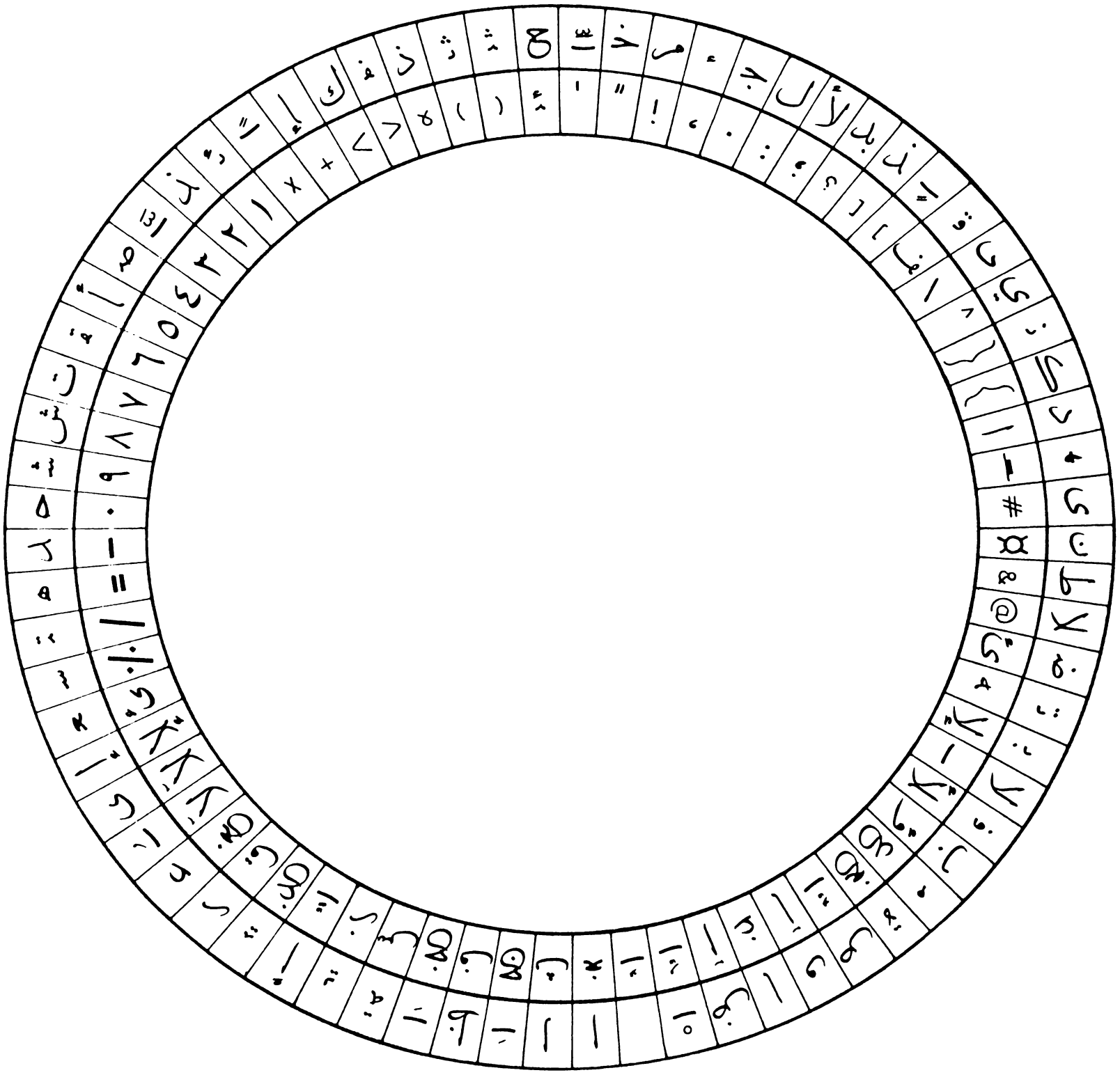


Figure (3) : Standard daisy wheel using the Koufi style.

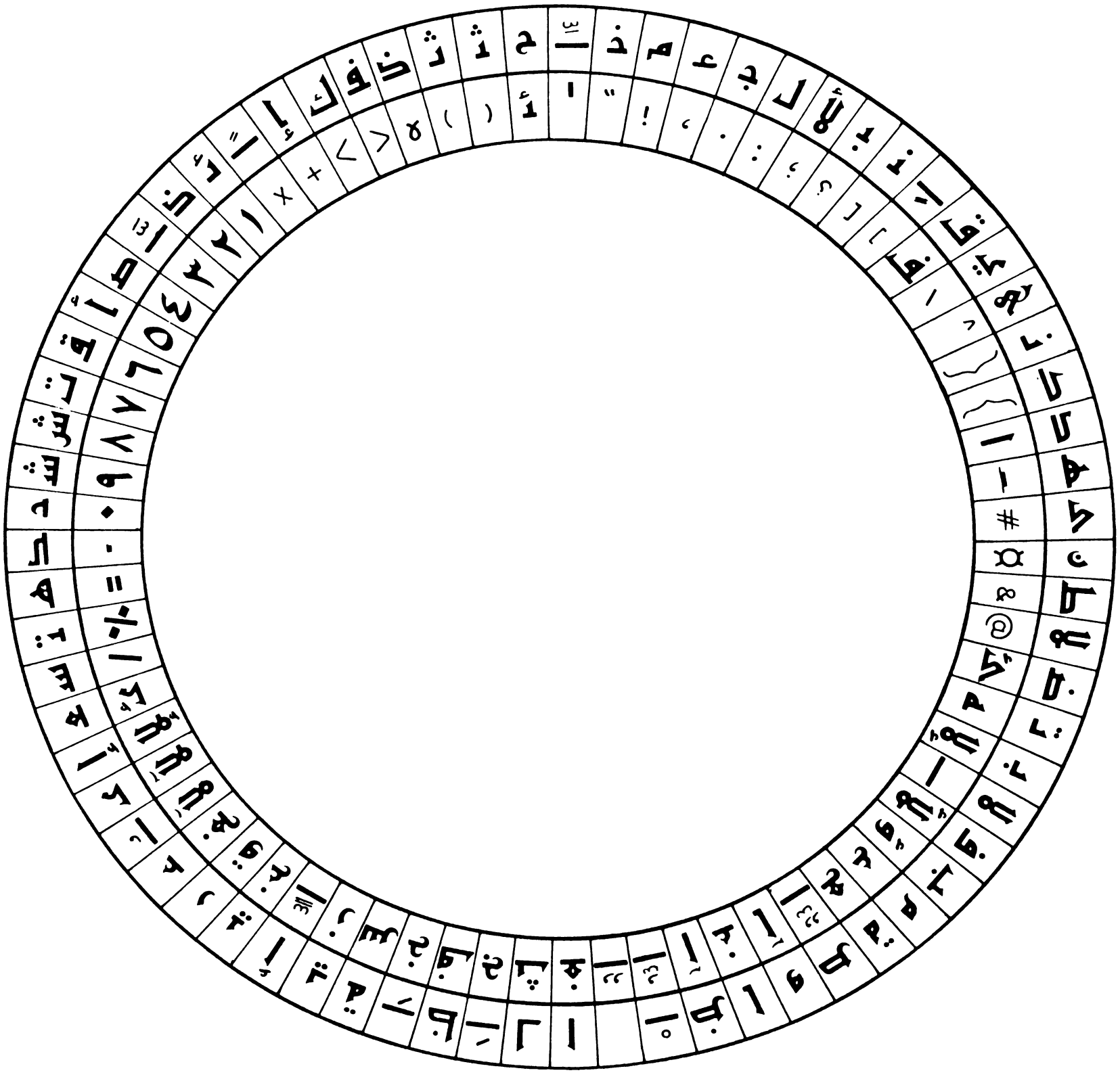
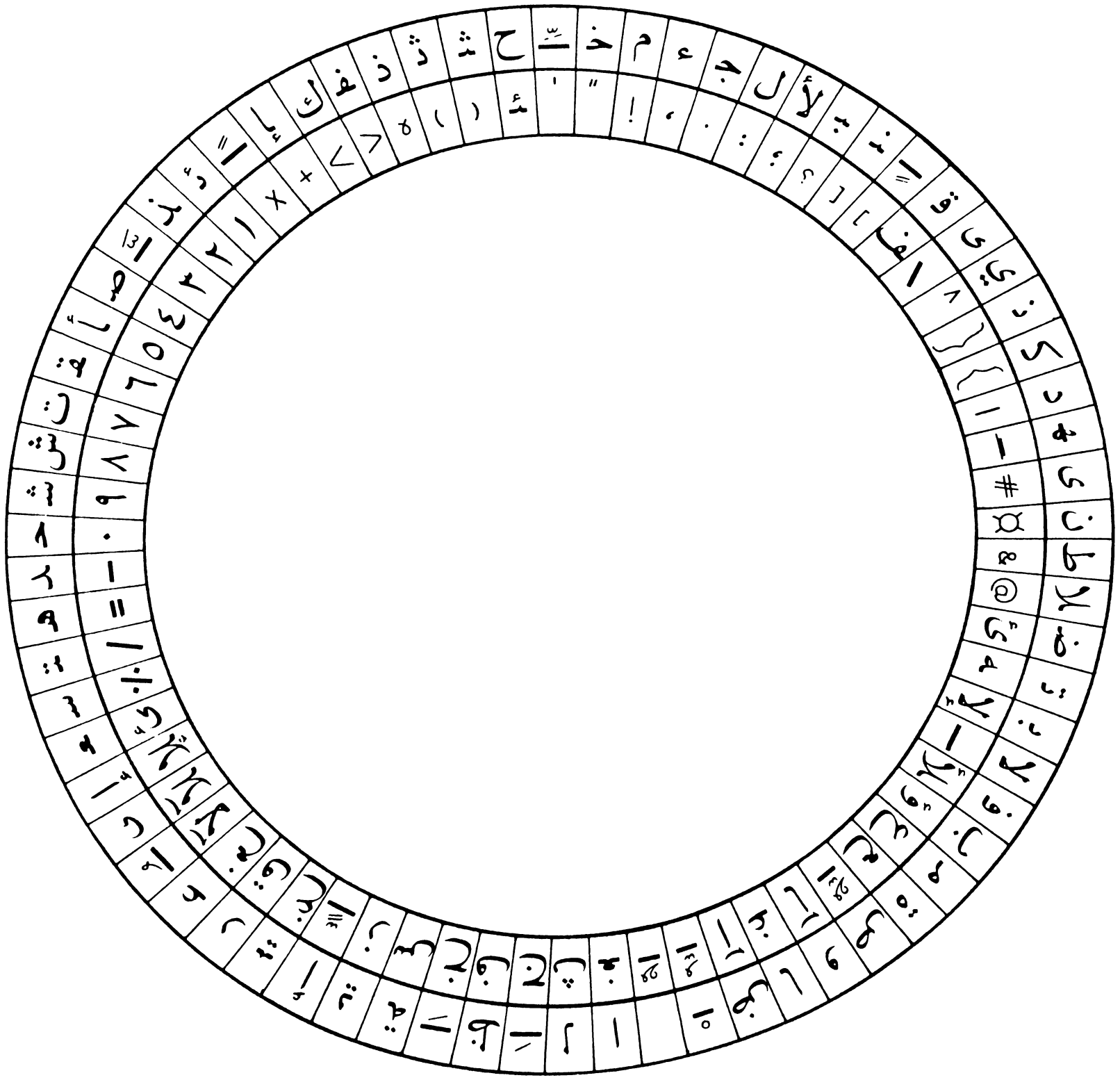


Figure (4): Standard daisy wheel using the Naskh style.



6 - *Design Considerations Of Arabic Peripherals In EDP Environment.*

So far, the general design issues for Arabic peripherals were discussed, and highlights were fo - cused on the quality of the output and on the statistical approach for optomizing the performance. The use of Arabic script in an EDP environment is examined and some further design considerations are presented. It will be shown that the statistical properties of the Arabic letters used in EDP environment are different from those obtained using ordinary texts. Considerations for the choice of character repertoire for EDP is also discussed, provided that the repertoire at the output of the contextual analysis remains standard, see table (1). These considerations are then applied in the design, realization and optimization of an Arabic belt printer.

6-1 EDP versus office automation environments

Office automation systems are expected to be one of the most important computer applications in the Arab world in the near future. By their nature, they imply the use of quality Arabic script since the resulting documents are submitted to public use and to top management. But on the other hand, these systems are less critically affected by performance of the I/O devices, especially with respect to their speed. On the other hand, the EDP environment is relatively well established throughout the Arab world with its traditional applications such as: payroll, inventory control, accounting, invoicing , personel,...etc. These types of applications are in fact much more tolerent with respect to the quality of Arabic printout but much more insistent on the performance side since huge amounts of reports are issued by them.

Another important distinction between these two environments is the extensive use of numerals in the EDP environment in comparasion with office automation where letters are largely dominante.

Though the essential design issues for Arabic peripherals remain the same in the two environments, some important design considerations are quite different. In an EDP environment priority is given to performance over calligraphic quality. This priority should however preserve the legibility of the text and traditional beauty of the Arabic calligraphy. Thus , some compromise should be realized.

The performance of some peripherals depends on the statistics of the Arabic letters. But most of the studies, carried out to determine some of their statistical properties, used samples of Arabic texts extracted from literary texts or newspapers. The nature of these samples is quite different from that encountered in an EDP center, especially in the two following points:

1 - *They do not take into account the statistics on the use of numerals and special characters.*
2 - *Proper names are much more frequent in an EDP environment than it is in the case of literary texts.*

Thus, it is essential to determine the statistical properties of the Arabic letters for EDP environment using a representative sample reflecting the real use of these letters in this particular environment.

6-2 Frequencies of characters used in an Arabic EDP center.

The main concern in the present paragraph is to determine, with some precision, the frequency of occurrence of the different characters used in an Arabic EDP environment. This should cover:

- *a- The Arabic letters (with their different shapes).*
- *b- The numerals.*
- *c- The special characters.*

Construction of the sample

The first step in any statistical study is to build the representative sample to be used in order to obtain staistics reflecting the general case. IN the present study this sample was constructed using I/O data related to a collection of applications which are quite different in their types and which belong to 5 different sources namely: SSRC, Ministery of work, FIGEH water autority, ministery of Housing and SYRONICS company.

The following table shows the composition of this sample and the size of data used in each application:

Application type	*Size of data*
Contract follow up	*3500*
Vehicle management	*400*
Health security system	*3500*
Public housing	*20200*
Inventory control	*2635*

Office automation	*11605*
Payroll	*4200*

The size of the data in each application is given in records in order to eleminate the differences in the number of bytes related to different I/O types (entry, reports, screen).

Statistics by type of I/O

The use of data in any application of an EDP environment is assymetric with respect to input and output. In fact most applications require data entry through forms systems. Thus, what is displayed on the screen (the text of the form) is repetative and uses generally few numerals, while the entered data itself is, in most cases, basically numeric. On the other hand tables and reports that are issued by the application are a mixture of data and texts.

So, in order to obtain statistics that can be used rationally to design each type of peripheral devices (keyboard, printer, display,...) statistics should be classified according to the type of device in at least two categories:

a- Data input (for keyboard).

b- Data output (printer).

Another important motivation for such classification is the fact that the repertoires of Arabic characters for input and output are quite different due to the difference of the shapes a single character might have.

Results

The following tables give the frequency of occurrence of the different characters in the above mentioned classes of statistics. Table (9) is for data entry through keyboard. The total size of data used to construct this table is:

- *alphanumeric: 1,172,000 character*
- *alphabetic only: 563,000 letter*

while table (10) gives these statistics for data output on a printer.

The corresponding size of the sample is:

- *alphanumeric: 3,697,000 character*
- *alphabetic: 1,192,000 letter*

In order to compare the above results with previous studies, it was necessary to eleminate the special characters as well as the numerals, and consequently to modify the statistics to take into

FREQ.	CHAR.	FREQ.	CHAR.	FREQ	CHAR.	FREQ	CHAR.	FREQ.	CHAR.
1.943	ب	0.006	ا	0.004	آ	0.023	أ	4.929	ا
0.396	خ	2.110	ح	0.760	ج	0.126	ث	0.645	ت
1.765	س	1.344	ز	3.230	ر	0.028	ذ	3.480	د
0.073	ظ	0.555	ط	0.183	ض	0.506	ص	0.700	ش
0.724	ك	1.060	ق	1.062	ف	0.171	غ	1.716	ع
2.704	ة	0.763	ه	2.144	ن	5.894	م	2 350	ل
0.167	ى	0.092	د	4.581	ي	0.016	و	1.786	و
23.26	·	0.000	لا	0.00	لآ	0.001	لا	0.171	لا
1.559	٥	1.855	٤	2.728	٣	3.541	٢	10.162	١
0.003	—	4.370	٩	1.226	٨	1.535	٧	1.584	٦
0.005	)	0.000	٪	0.00	#	0.0	"		١
0.007	—	0.006	ر	0.001	+	0.002	*	0.005	(
0.000		0.00	؛		:	0.02	/	0.01	.
		0.00	ﷺ	0.00		0.0	<		=

Table (9): Frequency of characters for data input in Arabic EDP .

FREQ	CHAR.	FREQ	CHAR.	FREQ	CHAR.	FREQ	CHAR.	FREQ	CHAR.
0.005	آ	0.032	أ	0.123	أ	1.738	ا	2.085	ا
0.218	ب	0.976	ة	0000	إ	0.01	إ	0.004	آ
0.437	ج	0.030	ث	0.070	ؤ	0.358	ت	0,842	ت
0.042	خ	0.259	خ	0.132	ح	1.026	ح	0.897	ج
1.269	س	0.635	ز	1.98	ر	006	ذ	1.83	د
0.070	ص	0.396	ص	0001	ش	0.385	ش	0.096	س
0.609	ع	0.096	ظ	0.441	ط	0.022	ض	0.103	ض
0.087	غ	0.056	غ	0.099	ع	0.059	ع	0,403	ع
0.806	ق	0.222	ف	0.604	ف	0.005	غ	0.025	غ
0.023	ل	1.891	ل	0.081	ك	0.385	ك	0.232	ق
0.177	ه	0.631	ن	0.732	ن	0.561	م	2.700	م
0.025	ؤ	1.350	و	1.480	ة	0.152	ه	0.185	ه
0.000	ئ	0.113	ئ	0.200	ي	0.478	ي	2,120	ي
0.001	لآ	0.045	لأ	0.329	لا	0.083	ى	0000	ى
1.452	٣	1.797	٢	4.460	١	10.276	٠	0000	لإ
1.203	٨	0.915	٧	1.043	٦	1.740	٥	1.283	٤
0.000	#	0.00	ء	5.059	١	0.056	—	2.101	٩
0.002	+	0.004	*	0.008	(	0.008	)	0.009	٪
0.06	:	0.430	/	0.309	·	19.826	—	1.248	ر
0.000		0.00	<	2.60	=	0.000	>	0.000	؛
								0.000	ﷺ

Table (10): Frequency of characters for data output in Arabic EDP.

account this elemination. The relative frequencies of occurrence of the Arabic alphabet (as seen in a EDP center) are presented in table (7) versus the values obtained by an Iraqi and à Moroccan study.

This table shows the following points:

a- There is a clear difference between the frequencies based on input data (column 1) and those related to output(column 2) though the general tendencies remain the same.

b- An important divergence existe between the values obtained in the present study and the results of previous ones based on texts especially with respect to highly frequent letters like (ل، ا، ؤ)

c- The present result are closer to Iraqi results.

The difference between the recent result and previous ones is in our opinion due to the following reasons:

The structure of data in EDP environment is statistically different from ordinary texts because of its fragmentation.

The presence of a huge amount of proper names in EDP environment with exceptional repetition of certain names like (احمد، محمد) disturb the statistics of the language. In fact, one can notice that the frequencies of (ج، د، م) are quite greater than ordinary frequencies. A simple calculation shows also that the difference between normal frequencies and the present one is 2 times greater for (م) than for (ج، د) and this corresponds well to the case of highly frequent names of (محمد)

SSRC	Marco	Iraq	input	output	chracte
0.38	0.20	0.47	0.04	0.11	ء
0.15	0.00	0.11	0.01	0.02	آ
2.38	1.5	1.79	0.05	1.44	ا
0.12	0.1	0.08	0.03	0.07	و
1.11	0.1	0.59	0.01	0.04	إ
0.38	0.4	0.56	0.10	0.34	ئ
15.00	16.5	11.93	10.25	11.35	ا
3.59	3.2	4.17	4.04	3.54	ب
2.73	3.1	4.80	5.60	4.39	ة
4.42	5.0	4.19	1.34	3.56	ت
0.66	0.4	0.63	0.26	0.30	ث
1.25	1.4	1.97	1.85	3.95	ج
1.79	1.8	2.5	4.38	3.44	ح
0.84	0.7	1.12	0.82	0.88	خ
2.72	2.8	3.25	7.20	5.43	د
0.99	0.8	0.63	0.05	0.18	ذ
4.17	3.9	6.29	6.70	5.88	ر
0.51	0.5	1.25	2.70	1.88	ز
2.16	2.2	3.11	3.67	4.05	س
0.79	0.9	1.33	1.46	1.32	ش
0.98	0.9	1.46	1.05	1.38	ص
0.68	0.5	0.93	0.38	0.37	ض
0.99	1.1	1.81	1.15	1.3	ط
0.28	0.2	0.29	0.15	0.28	ظ
3.49	3.4	3.61	3.57	3.47	ع
0.44	0.3	0.86	0.35	0.51	غ
2.75	2.5	2.76	2.2	2.45	ف
2.28	2.2	3.20	2.2	3.07	ق
1.93	2.1	1.87	1.50	1.38	ك
11.80	12.1	8.36	4.80	6.70	ل
6.01	6.0	6.66	12.30	9.67	م
5.19	4.9	4.42	4.46	4.04	ن
3.39	2.9	1.53	1.58	1.52	ه
5.76	5.6	3.99	3.72	4.00	و
0.91	0.8	0.71	0.34	0.25	ى
6.98	7.2	6.76	9.50	8.30	ي

Table (7): Comparaison of results with Iraqi and Marocan results.

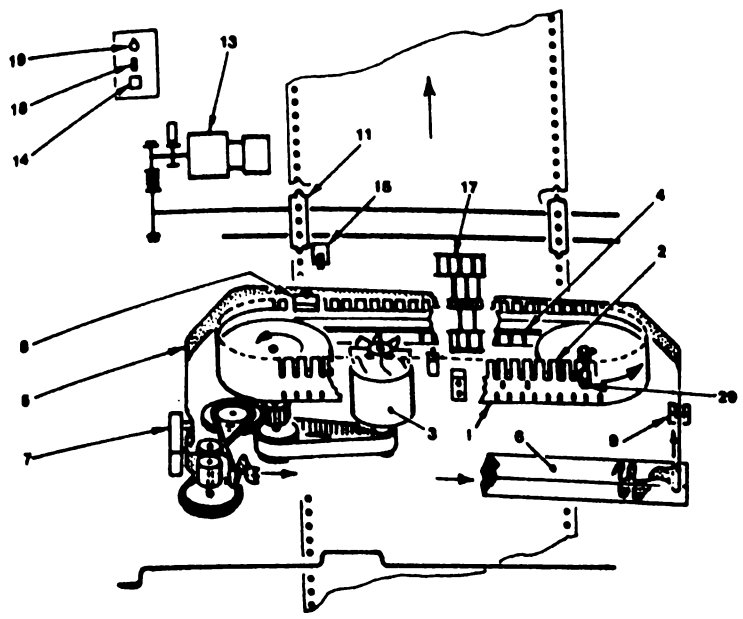
7 - Performance Optimization of a Bilingual (Arabic / Latin Belt Printer)

In opposition to serial printers, which print the output stream of characters sequentially one by one, line printers function on a line by line basis. The line to be printed is stored in a line buffer which contains a raw of storage cells, each one of them corresponds to a single position in the line. Then the content of the buffer is printed according to some order depending on the current position of the printing mechanism as well as the content of the buffer itself.

7-1 Operating principles of a line printer

The printing mechanism of a line printer figure(5) is mainly composed of the following elements:

- *A row of hammers, each one of them assigned to a printing position on the line.*
- *A rotating belt for (or chain) on which all printable characters are engraved side by side. The belt rotates continuously and passes between the hammer raw and the paper.*
- *A printing and synchronization logic.*
- *An inking mechanism.*



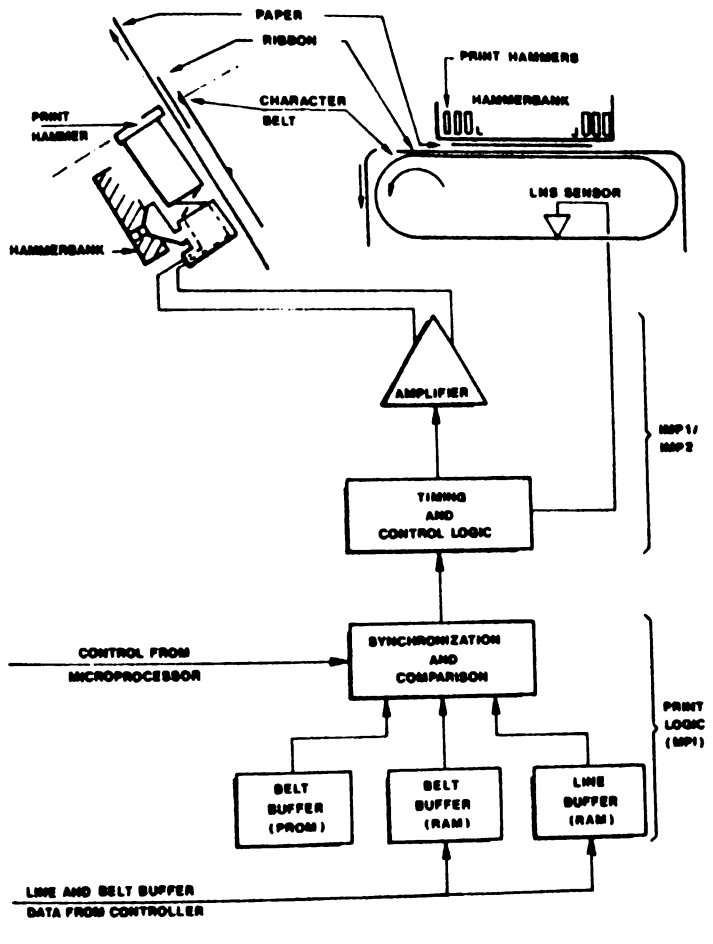


Figure (5): Printing mechanism of a belt printer.

As the belt rotates, the different characters engraved on it pass sequentially in front of each hammer. The printer's logic compares at each moment the content of each storage cell with the character passing at that moment in front of the printing position assigned to this cell. If they are identical, the logic will cause the hammer to impact the paper and consequently to print the character at that particu-ar position. Many hammers may be activated at the same time resulting in printing more than one character in parallel. Thus, the order of printing the characters in the line is variable and depends mainly on the position of the belt at the start of the line printing. When all the positions of the line are printed, the printing logic will cause a line skip and the procedure is repeated.

It is clear, from the above discussion, that the hammer will wait until the proper character shape passes in front of it before printing the character at that position. This waiting time represents the main delay required for typing the characters and is the primary element which determines the real speed of the printer. Decreasing the mean waiting time might increase considerably this speed. The theoritical speeds indicated by the vendors are no more than upper limits that can be obtained in using well designed belts, and they can drop to half or even a third of that value if improperly designed belt are used.

7-2 Design issues of a bilingual Arabic / Latin printing belt(or chain)

There are three main criteria in designing a printing belt for bilingual Arabic/Latin printers:

- *a- Choosing the set of shapes to be implemented on the belt (characters repertoire).*
- *b- Determining the number of occurrences of each shape on the belt in view of enhancing the printer's speed.*
- *c- Defining the calligraphic shape of each letter.*

The first and the third consideration deal with the legibility of the printed text, and aim at improving it, taking into account the limitations imposed by the technologies. These points are most important in the context of Arabic script because of the large number of character shapes in the classical Arabic caligraphy and the complexity of these shapes.

The second criterion is the key for enhancing the speed of the printer. However an important interaction between the first and the second criterion leads to some difficulties. In fact, enhancing the speed requires increasing the number of occurrences of the characters on the belt. Since the total number of occurrences on a given belt is fixed, the number of shapes have to be limited in order to maintain a reasonable speed, and this would have negative effects on the ligibility of the text. Some compromise is to be admitted.

7-3 Choosing the character repertoire

As indicated in the previous paragraph, a

compromise between the quality of the calligraphy and the performance of the I/O device is to be realized when designing an Arabic peripheral for EDP environment. Increasing the speed of the printer requires decreasing the size of the output repertoire. The repertoire of high quality Arabic printout includes about 90 different shapes, but in order to get the full benifit of this large number of shapes, variable width characters are necessary. However belt printers have fixed width characters and this implies important distortion of some shapes. In particular, medial shapes like:

بـ ، تـ ، ثـ ، نـ ، يـ

need smaller width than the other characters, and some other like: سـ ، شـ ، صـ ، ضـ require larger size. As a result of detailed study on the possibility of reducing the number of shapes of the repertoire without influencing the quality of the calligraphy the following repertoire was adopted corresponds to 73 shapes and preserve the essential properties of the Naskhi style.

ا	ـا			ظ			
أ	ـأ			ع	ـعـ	ع	ع
آ	ـآ			غ	ـغـ	غ	غ
إ	ـإ			ف		ف	
ب	ـب			ق		ق	
ت	ـت			ك		ك	
ث	ـث			ل		ل	
ج	ـج			م		م	
ح	ـح			ن		ن	
خ	ـخ			هـ	ـهـ	هـ	هـ
د				ة			
ذ				ؤ			
ر				و		و	
ز				ي	ـيـ	ي	ي
س				ى		ى	ى
ش				ىٔ			ىٔ
ص				لا			
ض				لا			
ط				لا			
ل				لا			
				ء			

7-4 Optimization of the printing speed

The main idea is to determine the number of occurrences on the belt of each character shape in order to obtain a minimum mean waiting time for printing a character taking into consideration the frequencies of each character shape.

Let's denote by:

- Nb The total number of all character occurrences on the belt (the number of positions on the belt)
- Nc The number of characters of different shapes to be implemented on the belt (the size of the repertoire).
- Nl The line length in characters (normally 132).
- i An index to indicate the position of a character in the line.
- j The jth character; (of the repertoire).
- Nj The number of occurrence of the character j on the belt;
- w The rotating speed of the belt.

We have seen in a previous paragraph that the characters of the line are not printed according to their sequential order, but in function of the position of the belt at printing time. Thus, the conditional probabilities of sequences of characters are of minor importance as compared to the case of serial printers (daisy wheel for example).

The most important part of the printing delay of a character is due to positioning the right character shape at each printing position. The line can be considered as a raw of Nl independent printing positions which can be occupied by any character. The mean time tij required to position the proper character j at the position i can be given by:

$$tij = \frac{1}{2w} \cdot \frac{1}{Nj} = \frac{1}{2wNj} \qquad 1$$

This formula postulates that the Nj occurrences of the character shape are uniformally distributed on the belt so that the mean waiting time is equal to half the time elapsed between two consequitive occurrences of the character.

Let's denote by Pj the probability that the jth character occupies the position i. the mean delai ti to print a character at the position i becomes:

$$ti = \sum_{j=1}^{j=Nc} pj\ tij = \frac{1}{2w} \sum_{j=1}^{j=Nc} \frac{pj}{Nj} \qquad 2$$

Maximizing the speed of the printer requires minimizing each ti or equivalenly minimizing the sum S:

$$S = \sum_{j=1}^{j=N} \frac{Pj}{Nj} \qquad 3$$

The values Pj are independent of the belt design and are at the same time the frequency of the letter j. Thus the optimization problem could be stated as follow:

Given the values of Pj of each character shape, and given the repertoire of character shapes find out all Nj values for $1 \leqslant j \leqslant Nc$ which minimize the sum S and respect the following conditions:

$$\forall\ j\ ;\ Nj \geqslant 1 \qquad 4$$

$$S = \sum_{j=1}^{j=Nc} Nj = Nb \qquad 5$$

Equation 4 indicates that at least one occurrence of each printable character shape should exist, while equation 5 means that the total number of occurrences should be equal to the number of positions on the belt. The following algorithm gives the solution to the above stated optimization problem.

7-5 Optimization algorithm

The idea of the algorithm is to begin by assigning one occurrence to each character shape. Then to proceed in steps. Each step consists in adding a new occurrence. This occurrence is assigned to the character that would minimize the sum S at this particular step. This proceedure continues until Nb is acheived.

The steps of the algorithm are the following:

Step 1 - Assign 1 to each Nj to satisfy equation 4

Step 2 - Calculate the sum $S' = \sum_{J=1}^{j=Nc} Nj$

Step 3 - If the sum $S' = Nb$ then print results and stop else continue.

Step 4 - Calculate K = j which maximize :

$$\left(\frac{Pj}{Nj} - \frac{Pj}{(Nj)+1} \right)$$

Step 5 - Increment Nk by 1

Step 6 - Go to step 2

This optimization proceedure was applied to design an Arabic/Latin belt for the PR54 BULL printer. The number of different characters which can be implemented on the belt is 240. Since the optimization is made to acheive maximum speed in printing Arabic only one occurrence of each latin character is retained this corresponds to

- 26 character
- 10 digits
- 6 special characters

The size of the Arabic character repertoire including special characters and numberals is 103. Thus:

- Nb = 198
- Nl = 132
- Nc = 103

The following table gives the result of applying the algorithm. Column 2 gives the frequency of the character while column 3 gives the number of occurrences on the belt.

3	2	1	3	2	1	3	2	1	3	2	1	3	2	1
1	0.01	آ	1	0.03	ٱ	1	0.12	٦	3	1.74	ا	3	2.08	ا
1	0.22	ب	2	0.98	ب	1	0.00	ٮ	1	0.01	ٮ	1	0.00	٦
	0.44	ج	1	0.03	ث	1	0.07	ث	1	0.36	ت	2	0.84	ت
1	0.04	خ	1	0.26	ح	1	0.13	ح	2	1.03	ح	2	0.89	ج
ئ	1.27	س	2	0.64	ز	3	1.98	ر	1	0.06	ذ	3	1.83	د
1	0.07	ص	2	0.40	ص	1	0.06	ش	2	0.39	ش	1	0.10	س
2	0.61	ع	1	0.10	ظ	2	0.44	ط	1	0.02	ض	1	0.10	ض
1	0.09	غ	1	0.06	غ	1	0.10	ع	1	0.06	ع	2	0.40	ع
2	0.81	ق	1	0.22	ف	2	0.60	ف	1	0.01	ع	1	0.03	غ
1	0.02	ل	2	1.98	ل	1	0.08	ك	2	0.39	ک	1	0.23	ق
1	0.18	ه	2	0.63	ن	2	0.73	ن	2	0.56	م	4	2.70	م
1	0.03	و	3	1.35	و	3	1.48	ة	1	0.15	ه	1	0.19	ه
1	0.00	ى	1	0.11	ى	1	0.20	ي	2	0.48	ي	3	2.12	ي
1	0.00	لا	1	0.05	لأ	1	0.33	لا	1	0.08	ى	1	0.00	ى
3	1.45	٣	3	1.78	٢	5	4.46	١	7	10.27	٠	1	0.00	لا
3	1.203	٨	2	0.95	٧	٤	1.04	٦	3	1.75	٥	3	1.28	٤
1	0.00	#	1	0.00	٬	5	5.06	١	1	0.06	-	3	2.10	٩
1	0.00	+	1	0.01	*	1	0.01	(	1	0.01	)	1	0.01	٪
7	10.06	:	2	0.43	/	1	0.30	.	10	19.8	-	3	1.25	،
1	0.00	?	1	0.00	<	4	2.6	=		0.00	>	1	0.00	؛
												1	0.00	@

Table (8)

7-6 Defining the calligraphy of characters

The objective of this paragraph is to define a methodology that would permit the reproduction of calligraphic shapes of characters engraved on the belt so that their shapes remain as close as possible to their original ones in the traditional Arabic sytles taking into account the limitations imposed due to the use of some engraving technology. The main technology used for engraving the characters on a belt is based on a chemical proceedure which imposes some limitations on the dimension and on the arrangement of the segments of the characters. One should mainly respect certain minimal values which are the following:

a- The minimal width of a segment.

b- The minimal distance between two segments.

c- The minimal curvature of the segment.

d- The minimal curvature at the segment end.

e- The minimal area of the character surface.

In addition to the previous limitations an additional important one is due to the fixed width of the characters in the case of a belt printers and also to the well known limitation due to the necessity of connecting characters.

The shapes of the characters of the belt are normally prepared on special sheets within a frame defining the edges of character area. these shapes are then modified in order to produce the maskes necessary to the chemical proceedure. The phase which concerns the designer of an Arabic belt is only the first phase, i.e, the preperation of the character shapes. It is recommended that the designer follows the following steps in preparing these shapes in order to guarantee the quality of shapes:

a- Prepare the special sheets with the frames in which the characters are to be drawn. Draw in this frame a horizontal line indicating the level of connection of characters.

b- Prepare a set of slides containing the shapes of all characters of a choosen Arabic style prepared according to standard rules.

c- Project on the sheet the shape of a character using the proper slide, then adjust the size of the projected character and its position so that its connections with neighbouring characters are situated at the correct connection level in the frame and, so that its size is convinient.

d- draw a preliminary shape of the character in copying the projected shape.

e- modify the shape of the character in order to satisfay all the requirements and limitations imposed by the engraving proceedure.

Figure (6) shows a specimen of the sheets used to prepare the character shape, while figure (7) presents example of the type of limitation imposed by the engraving proceedure for the belt designed to the printer PR54 of the company BULL:

- minimum thickness of segment = 0,20 m.m
- mean thickness of segment = 0,35 m.m
- minimum diameter of an inner curvature=0,35m.m
- minimum diameter of outer curvature = 0,20 m.m

The most difficult aspect of character shapes to be realized with these limitations is the three points of the characters ث،ش and also the final forms of characters ي . One more dilicate problem is to realize a small circular opening (less than 0,35) in the characters like و etc. and that distort the classical form of these characters. Figure (9) presents the output of the Arabic repertoire printed by the realized belt.

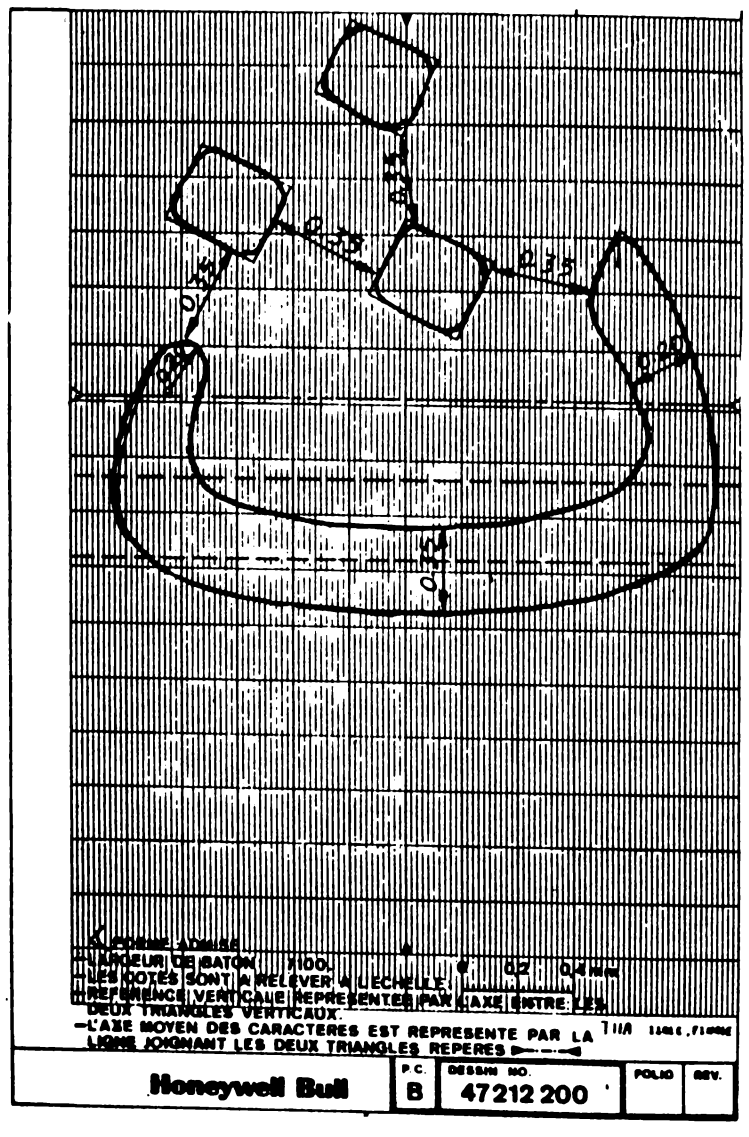


Figure (6)

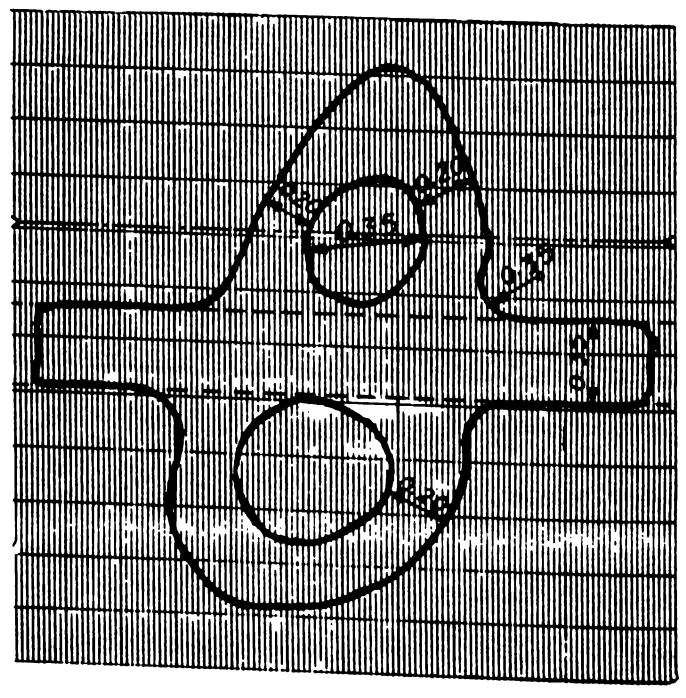


Figure (7)

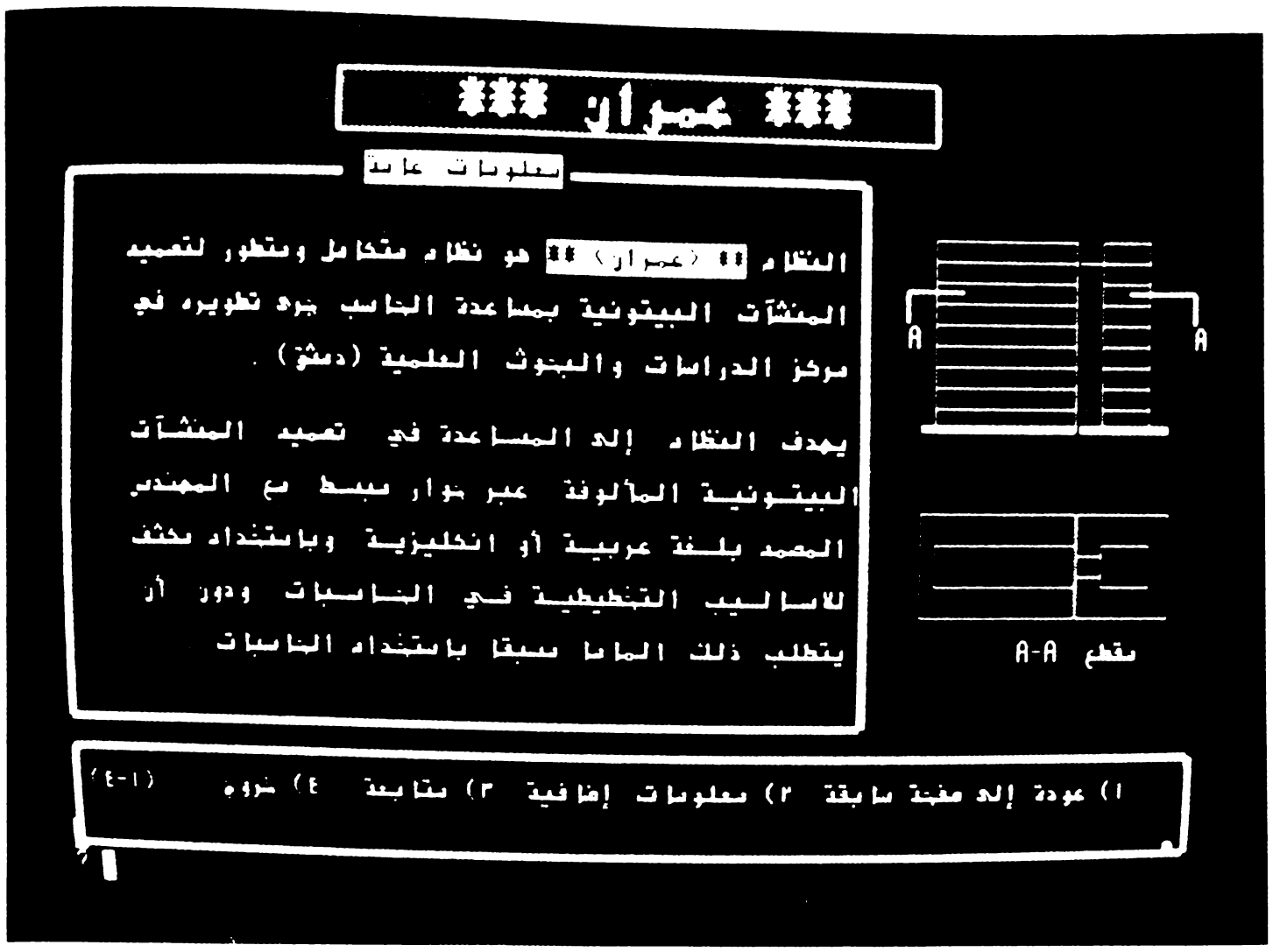


Figure (8 -a): Example of the screen of a display terminal using 10x16 dot matrix in the Koufi style.

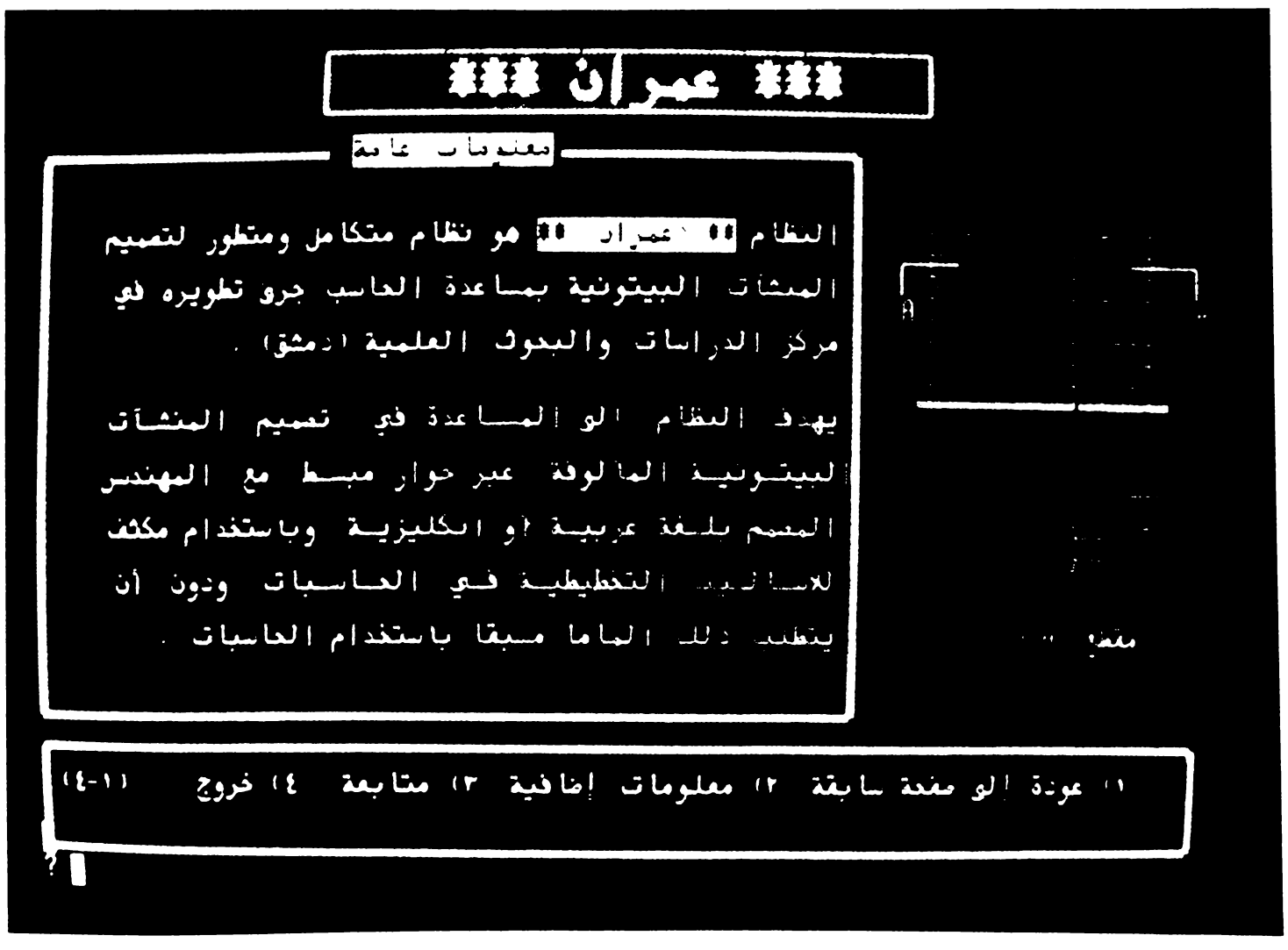


Figure (8 -b): Example of the screen of a display trminal using 10x16 dot matrix in the Naskh style.

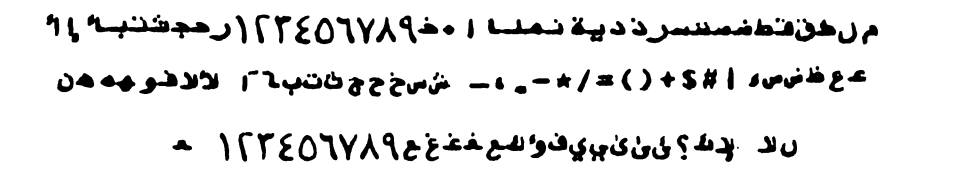


Figure (9)

8 - Conclusion

In this study we have presented the computer I/O of Arabic script as a simple I/O system. We proposed the addition of 5 diacritic signs to the ASMO 449 standard. We also proposed a standard for Arabic character repertoire at the output of contextual analysis algorithm. Basis criteria for the design of Arabic I/O peripherals are given with examples on the design of a daisy wheel, and a printer belt. Figure (8) shows an example of a display terminal using the Koufi and the Naskh styles using a matrix of 10x16 dots. Keyboard layout design is discussed in reference 9

Acknowledgement

The authors wish to acknowledge the very useful participation and contributuions of Mr. M. Bawab on the statistics of the Arabic language, Mis. N. Idlebi on the daisy wheel program and Mr. Z. ZARZOUR and Mr.W.Chouhayed for the caligraphic presentation of Arabic characters. Also thanks to Mr. G. Jala who developed the statistics on Arabic characters in EDP environment.

References

(1) *SHALIE R.A., "Vowelized Text on an Arabic Terminal", workshop on Computer Processing of the Arabic Language. KUWAIT 1985.*

(2) *DEWACHI A., "Problem Areas in the Treatment of Arabic in Hardware and Software Systems", Workshop on Computer Processing of the Arabic Language, Kuwait 1985.*

(3) *KHAYAT M.G., "Printing Arabic Text Using Dot Matrix Printers", Workshop on computer processing of Arabic Language, Kuwait 1985.*

(4) *MOUNAJED M.B., "Use of Arabic Script in Computer and Communication Media", Arab School on Science and Technology: Applied Arabic Linguistics and Signal and Information Processing. Rabat 1983.*

(5) *KOTAB S., HOSNI Y., "Assessment of Microcomputer Arabization Efforts". Workshop on computer processing of the Arabic Language, Kuwait 1985.*

(6) *GHAZAL, A.A. "L'Alphabet Arabe et les Machines", Proceedings of the Arab School on Science and Technology: Applied Arabic Languistics and Signal and Information processing. Rabat 1983.*

(7) *ABOU-AL-HAYJA, A.; and IDRIS,F., "Evaluation of proposed Arabic Keyboard Layout". Internal Report (in Arabic), Yarmouk University, Jordany 1984.*

(8) *MRAYATI, M., "Statistical Studies in Arabic Languistics", Proceeding of Arab School on Science and Technology. Zabadani-Syria 1985.*

(9) *IDLEBI, N. and MRAYATI, M. "Design of Arabic Keyboards Layout Based on Statistical Properties of Arabic Characters", Proceeding of Arab School on Science and Technology. Zabadani-Syria 1985.*

Design of Arabic Keyboard Layout Based on Statistical Properties of Arabic Characters

N. IDLEBI and M. MRAYATI
Scientific Studies and Research Center
P.O. Box 4470
Damascus, Syria

I. Introduction

The importance of computer is increasing, and it is being used in different scientific and management fields. It is originally designed for the English language and to use it correctly in Arabic, it is useful to study its interfacing with the Arabic language for optimum efficiency. As the keyboard is the main device for entering data, it is profitable to study it to facilitate, on the one hand, the work of its user and, on the other hand, the efficiency of data input.

The layout of Arabic characters on computer keyboard varies considerably with the company producing it. Up until now, no standard arabic keyboard has been developed. Dr. A. Abou Al-Hayja, in a study achieved at the Yarmouk University, evaluated different proposed keyboard layouts.

In this study two methods for distrubuting the set of arabic characters on the keyboard are considered using an optimization algorithm:

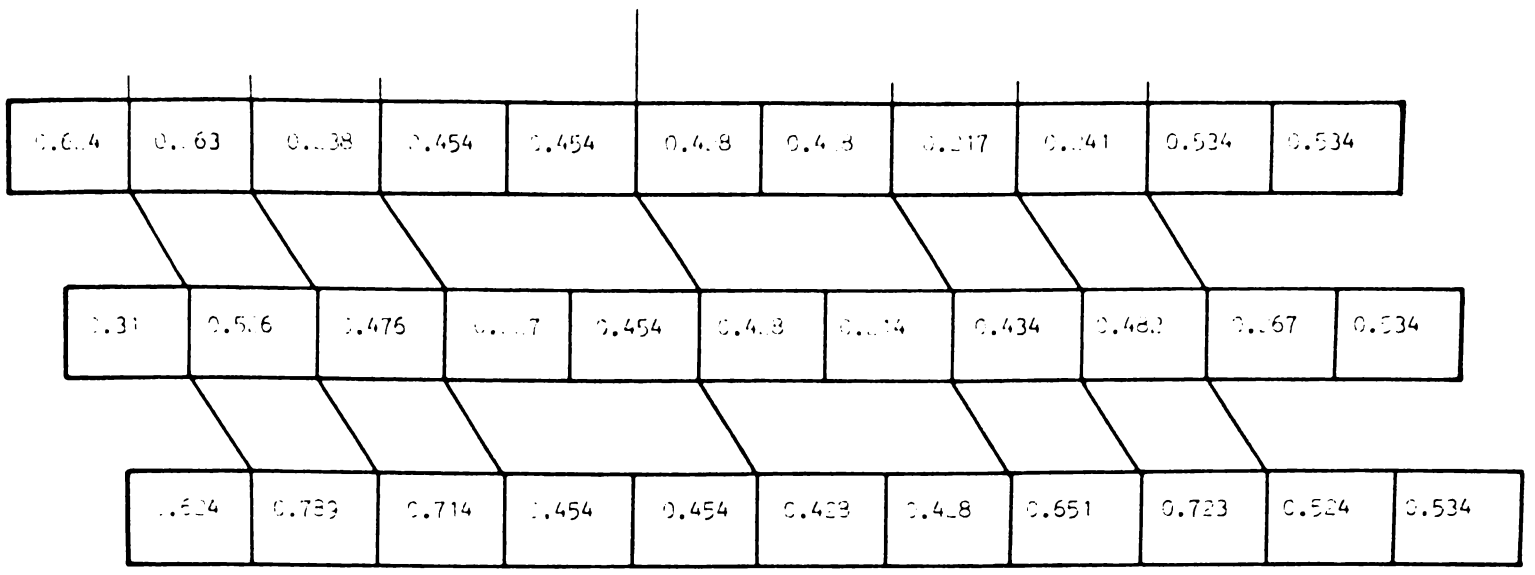
1 - The layout of arabic keyboard for mono-lingual applications.
2 - The layout for bilingual applications, which is based on the distribution of the arabic characters on the keys specified for latin ones, while conserving the position of numerals and special symbols keys.

II. Design Criteria

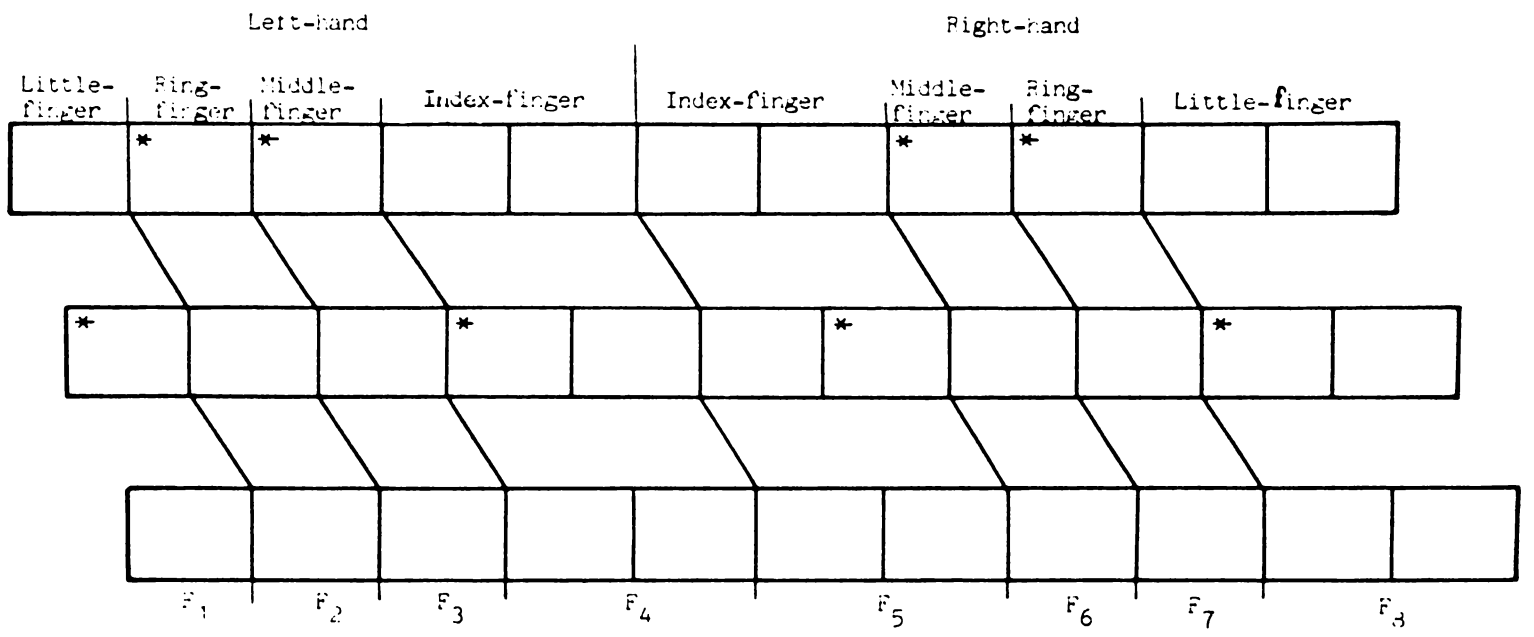
This study aims at the distribution of the ASMO 449 Arabic characters (letters and diacritics) on 33 keys for the first method, and on 26 keys for the second. Having reached this distribution the keys for numerals and special characters are added.

In order to achieve this aim several design criteria are employed and they are the following:

1 - The frequency of occurrence of Arabic characters in normal text are to be taken into account. The grapheme and bi-grapheme frequencies are found and used in an algorithm which searchs the optimal position for each character.
2 - The maximum number of Arabic characters has to be allocated to the unshifted key positions.
3 - The binary code corresponding to each character has no relation or restriction on the position of the character on the keyboard.
4 - The layout of the keyboard is designed regardless of that of the typewriter since the character set is completly changed.
5 - The time needed for each finger to type one key is given in figure (1). This time is essential in the optimization of the layout considered.
6 - The keys which are normally typed by each finger and their initial (rest) positions are illustrated in figure (2) and are used by the optimizing algorithm.
7 - It is worth noting that the eight-fingers are arranged according to their speed of typing as follows:
    a- index - finger of the right hand. (F5)
    b- middle- finger of the right hand. (F6)
    c- index - finger of the left hand. (F4)
    d- middle- finger of the left hand. (F3)
    e- ring - finger of the right hand. (F7)
    f- ring - finger of the right hand. (F2)
    g- little-finger of the right hand. (F8)
    h- little-finger of the left hand. (F1)
8 - The keyboard adopted in this study is the standard 48 key keyboard (not counting the control keys).



Time needed for typing each key in seconds.
For shifted position the time needed = 0.8 sec. always.
Figure - 1 -



* Initial position of the fingers
Figure - 2 -

III. Frequency of Occurance of Arabic Characters

Our study is based on statisitics deduced from samples of Arabic text composed of 100 000 Arabic letters including full diacritics. The frequency of occurrence of each grapheme and each bi-grapheme are found.

Since diacretics is not used always, and since many application does not need diacritics, we have weighted the frequency of occurrence of the diacr tic symbols by 30%. The resulting weighed . frequencies are used in the optimizing algorithm. Table (1)shows the frequency of occurrence of each character before and after weighing.

IV. The Optimizing Algorithm

Let us define the operation of typing a certain character as the loading of this character. Loading a character is determined by the frequency of occurance of this character multiplied by the time needed to type the key corresponding to it.

Frequency of Arabic Characters

Table - 1 -

Number	Character	Absolute Frequency	Weighted Frequency
1	ـَ	17,54	7,60
2	ـِ	10,35	4,49
3	ـُ	4,10	1,77
4	ّ	2,27	0,98
5	ـْ	0,69	0,30
6	ـً	0,40	0,17
7	ـٍ	0,40	0,17
8	ـٌ	0,58	0,25
9	ـٰ	0,20	0,089
10	ـّ	0,012	0,005
11	ـٍّ	0,03	0,01
12	ـٌّ	7,35	3,19
13	ـَّ	0,20	0,29
14	ى	0,51	0,74
15	ـا	8,49	12,28
16	أ	1,33	1,93
17	ـآ	0,62	0,90
18	آ	0,08	0,12
19	ف	1,55	2,24
20	د	1,53	2,21
21	ذ	0,55	0,79
22	ر	2,34	3,38
23	ز	0,28	0,41
24	و	3,23	4,66
25	ؤ	0,07	0,10
26	ب	1,98	2,87
27	ة	2,54	3,68
28	ت	3,80	0,55
29	ج	0,68	0,99
30	ح	0,99	1,44
31	خ	0,49	0,71
32	س	1,29	1,86
33	ش	0,44	0,64
34	ص	0,54	0,78
35	ض	0,40	0,58
36	ط	0,57	0,83
37	ظ	0,14	0,21
38	ع	1,92	2,78
39	غ	0,23	0,33
40	ف	1,51	2,18
41	ق	1,28	1,85
42	ك	1,06	1,53
43	ل	6,70	9,68
44	م	3,33	4,81
45	ن	2,84	4,10
46	ه	1,55	2,25
47	ئ	0,21	0,31
48	ي	3,98	5,75

Our purpose here is to optimize the operation of loading an Arabic text. This is achieved by distributing the 48 characters on the keyboard in such a way as to accomplish the following two results:

a - minimum loading time

b - equal load on all fingers

This purpose is realized in three steps as follows:

1 - Minimize loading characters which is defined mathematically by

$$L = \sum_{i=1}^{NBKEY} t(i) * f(i)$$

where: NBKEY Number of keys.

t(i) The time needed for a finger to leave its intial position, reach the key (i), type it, then return back to the intial position.

f(i) The frequency of occurrence of the letter present on key (i)

2 - Equalize the load on each finger.

We define the load on a finger F(k), k 1,2 . . 8 by $L_{F(k)}$

$$L_{F(k)} = \sum_{i \in N_{F(k)}} t(i) * f(i)$$

where: $N_{F(k)}$ The keys typed by the finger F(k)

Let M be the sum of differences between the loads of the different fingers namely:

$$M = \sum_{\substack{i=1,8 \\ j=1,8 \\ i \neq j}} |L_{F(i)} - L_{F(j)}|$$

We will minimize M to equalize the loads on each finger.

3 - To further minimize the text loading time, the time needed for a finger to move from one key to the next, is reduced by placing next to each character, the characters which have high frequency of occurrence with it. For this, we took into consideration the frequency of occurrence of the pairs of characters.

Let N be the sum of frequency of occurrence of pairs of characters (i,j) multiplied by a certain factor Alpha (i,j).

$$N = \sum_{\substack{i=1, NBCHAR \\ j=1, NBCHAR}} f(i , j) * Alpha (i , j)$$

where: NBCHAR The number of Arabic characters

f(i , j) The frequency of occurrence of pairs of characters (i , j)

Alpha (i , j) will be determined according to the places of two characters (i,j) on the keyboard as follows:

a - *Alpha (i,j) = 1, if the first character is presented on column (j) and the second one in column (j ± 1)*

b - *Alpha (i,j) = FIX 1, if the first character is typed by finger (k) and the second one by the finger (k ± 1)*

c - *Alpha (i,j) = 2 * FIX, if the first character is typed by finger k, and the second one by the finger (k ± 2)*

d - *Alpha (i,j) = 3 * FIX, if the two characters are typed by the same finger*

e - *Alpha (i,j) = 4 * FIX, if the first character is presented on shifted key position and the other on an unshifted key position or vica versa.*

Remark: In (1 - 2 - 3 - 4) it is supposed that the pairs of characters are both on shifted, or unshifted key positions.

To accomplish the optimization, we have minimized N.

Results

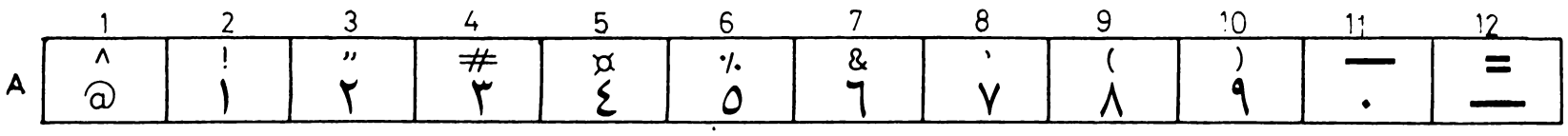
The problem is solved by using a program written in Pascal language. An intial solution of the Arabic keyboard layout is given to the program which calculate L_p, M_p and N_p. Then, an oriented change in the distribution is made, and a comparision between the new values of L_N, M_N, and N_N and the previous L_p, M_p and N_p is made. The change is either accepted or not according to the differences $(L_N - L_p)$, $(M_N - M_p)$, $(N_N - N_p)$. This operation is repeated several times until we achieve a near optimal solution.

Surely, at this stage we cannot obtain an optimal solution which has the smallest optimal values for L , M and N. Any change to the Arabic keyboard layout may give a smaller value for L, while M or N increase. The decision in this case depends on the relative decrease or increase.

Two solutions of the Arabic keyboard layout are made:

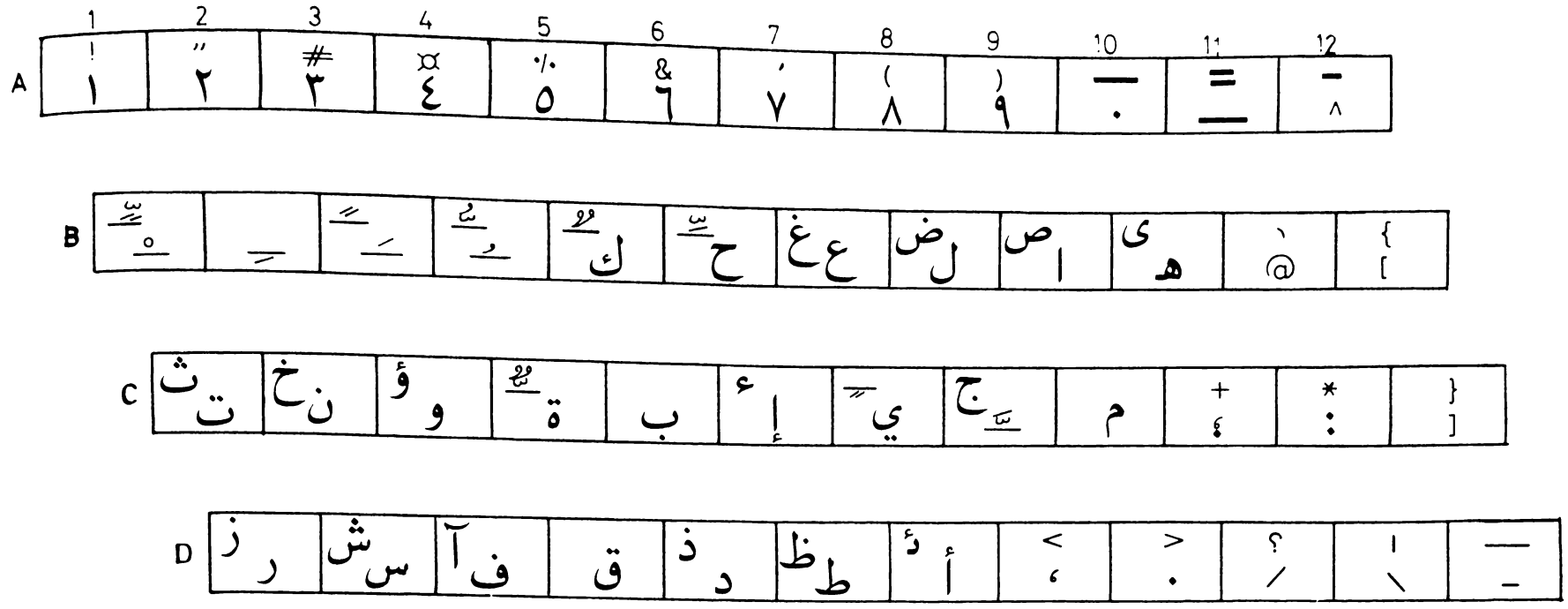
1 - *The layout of Arabic keyboard for mono-lingual applications figure (3).*

2 - *The layout of Arabic keyboard for bi-lingual applications figure (4).*



Layout of Arabic Keyboard

Figure - 3 -



Layout of bi-lingual keyboard based on 30% use of diacritics

Figure - 4 -

These two figures are given as an example on the use of this program. It is obvious that we can use this program to design a layout that takes into consideration other factors such as:

a - Lower percentage of use of diacritics namely weight less than 30%. It is interesting to note that even with 30% weighing the ـُ, ـَ, ـِ and ـْ are on unshifted key position.

b - A preferable position for certain characters, such as putting the ا and ل in position C7, C8 respectively.

Acknowledgement

The authors are thankful for Mr. M. Bawab for his cooperation in the work on Arabic character statistics, and for Mr. M. Farah for reading the proofs of the paper.

References

[1] *ABOU-AL-HAYJA-A and IDRIS.F, "Evaluation of proposed Arabic keyboard layout". Internal Report, Yarmouk Univesity, Jordany 1984.*

[2] *ECMA, European Computer Manufactures Association, "Arabic keyboard layout". Proposed ECMA Standard /TC1/84/.*

[3] *ASMO, "Arabic - Latin keyboard layout". The Arabic Standard Specification. 445. 1982.*

[4] *ASMO. Arab Organization for standarization and Metrology, "Standard specification for bi-lingual keyboard". Primary proposal, 17.8.1983.*

[5] *ASMO. "Standard specification for the keyboard", 663. 21.02.1985.*

[6] *ISO, "Keyboards for countries whose languages have alphabetic extenders - Guidlines for harmonization" . ISO 3243 - 1975.*

Standards for Arabic Characters in Informatics

MOHAMED AMIN AL-SALEH
ASMO TC 8
Damascus, Syria

1. GENERAL

Arab Standards & Metrology Organization (ASMO) formed in 1981, Technical Committee No.8, " Arabic Characters in Informatics ", to develop Arabic standards related to the processing of Arabic language in Computers and Telecommunication equipment. The technical secreteriat of this committee is the Syrian standards & Metrology Organization (SASMO). This committee issued the well known standard ASMO 449 dated October 1982 and called " Data processing - 7 bit coded Arabic Character set for Information Interchange ".

Arab Telecommunication Union's (ATU) " Data Transmission Committee" was formed to set up specifications on the equipment and protocols used for data transmission, and to solve Pan-Arab problems related to this area. This committee issued the well known standard ASMO 445 dated 1982 and called " Bilingual Teleprinter (Arabic/Latin) ".

These two committees have developped recommendations related to their common field " Teleinformatic", but no agreement have been reached concerning priorities and the way to proceed. In this paper we summarize the efforts undertaken by the Technical Committee No.8 towards standarization, and give a look at its future work.

2. ARABIC 7-BIT CODED CHARACTER SET FOR INFORMATION INTERCHANGE

General Assembly of ASMO adopted in October 1982 the Arabic Standard 449, " Data Processing - 7 bit coded Arabic Character Set for Information Interchange ". (1) A first ISO draft proposal for an international Arabic 7-bit code has been edited according to ASMO 449.

This standard specifies the properties of a coded character set using 7-bit binary codes for information interchange among different types of data processing equipments using the Arabic language. The set of graphic characters specified in the standard enables us under all circumstances to represent Arabic text whether it is totally vowelized, partially vowelized or unvowelized.

The Arabic language consists of a set of 44 letters and vowel characters. A letter is defined as an elemnet of the written language which represents an independent sound, and which joins other letters to make up words. Context analysis is the process of writing the correct shape of each letter, according to its position within the word.

The standard was developped including both a graphic character set and a control character set, which permits the usage of the Arabic coded character set as a seperate group from the Latin character set described in the international standard ISO 646.

3. BILINGUAL TELEPRINTER (ARABIC/LATIN)

In October 1982, ASMO General Assembly adopted the Arabic standard 445, "Bilingual Teleprinter (Arabic/Latin) ". This standard specifies the properties of a coded character set using 5-bit code for information transmission and reception, in both Arabic and Latin on the international telex network using the international alphabet No.2. It also specifies the minimal technical specifications required for operations in single mode Arabic and bilingual mode Latin

the design of a complete arabized computer.
- Allowance of additional characters in the future if required without predefining any positions number. These eventual positions are to be used by all nations whose languages use basically Arabic characters.

The table consists of 95 Arabic characters as shown (table 1) in addition to 30 control characters. The remaining 128 positions on the right hand side are provided for future extension as mentioned before.

7. 8-BIT ARABIC/LATIN CODE TABLE

Arabic and Latin character sets have been used simultaneously in computers for many years. For bilingual computers using 8-bit code tables, it is easier to have both character sets in the same code table.

A draft standard is now being prepared (6) and is expected to be adopted by ASMO General Assembly before the end of 1985. The main characteristics of the draft standard are the following:

- The set of graphic characters is intended for use in information interchange as well as data and text processing applications.
- It consists of 95 Latin characters and 50 Arabic characters identified as the Arabic-Latin Alphabet and specifies the coded representation of each of these characters by means of a single 8-bit byte. (table 2).

8. CONTEXT ALGORITHM, CHARACTER REPERTOIRE

It was agreed not to standardize a Context Algorithm, the development of which should be left individually to manufacturers To ensure that the presentation (reproduction) of received text is identical to its original, the directions and rules for such an algorithm have to be established. Furthermore, the Arabic character repertoire including all the shapes and styles which are intended to be used have to be determined. For practical implementation, two questions have to be answered namely: (7)

- What is the minimum of character shapes for different types of printers and styles.
- What is the "maximum" of the minimum in all these cases.

9. FUTURE LOOK

Above is a quick summary of Arabic standards which are either adopted or in the process of being adopted. This constitutes only a small fraction of standardizing the storage, processing, retrieval, and transmission of informations using the Arabic language.

The terminal is the most common type of equipment used for data entry and interactive retrieval of information. We beleive that the Arabic standards described above are not sufficient to manufacture a bilingual terminal in a standard way, which satisfies our national needs. There are still some problems to be answered before a "standard bilingual terminal" can be made. Some of the problems which have been defined include: Insertion of Arabic text into Latin text and vice versa, imaging of Arabic characters on the screen, and handling of numerical data and mathematical expressions (in the Arabic mode).

In a study (8) five points have been raised concerning the Technical specification for a bilingual terminal.

1- Is there any relation between Escape sequence and the coding mode? (an Escape sequence identifies the character set as well as the code in which the character set is coded).
2- What are the control characters used for insertion mode?
3- What is the control characters which identifies an 8-bit code table within a 7-bit or 8-bit environment ?
4- Define the relation between control functions used to identify the Arabic and Latin mode in a 7-bit and 8-bit environment.
5- Define the related keys and functions.

Arabic. It also gives general specifications of the keyboard, and the distribution of letters and signs on the keyboard. This distribution layout of Arabic characters has been established according to scientific criteria based on frequency analysis and ergonomy.(2)

4. CONVERSION BETWEEN 5-BIT AND 7-BIT CODES

A draft standard has been prepared which contains two conversion tables: (3) One for the conversion of 5-bit code to 7-bit code for imformation interchange, and the other table for the conversion of 7-bit code to 5-bit code . It is expected that ASMO General Assembly adopts this draft as an Arab standard in October 1985. This standard No. 584, when adopted, will enable the communications between computers and telex equipment, and the usage of information networks and 7-bit codes by telex.

5. BILINGUAL KEYBOARD

Since most data entry equipments have keyboards, there is an urgent need for standardizing an Arabic/Latin keyboard layout which defines the positions of all Arabic characters (as per ASMO 449) as well as all Latin characters on the keyboard, which have been already defined in ISO 2530.

Two different approaches were considered for designing the keyboard layout, one is based on the telex keyboard which was recently adopted in ASMO 445, while the other is based upon the traditional typewriter keyboard. These two approaches lead to quite different keyboard layouts. By majority, it was agreed to proceed with the keyboard layout which is based upon the traditional typewriter keyboard, and a draft No. 663 has been prepared. (4)

The main characteristics of the draft standard are the following:

- The maximum number possible of Arabic graphic characters is allocated to unshifted key positions, including three of the most frequent vowel characters: Fat'ha, Dammah, Kasrah. (Figure 1).
- All alphabetic Arabic graphic characters are allocated to keys in rows B, C, and D for increased accessibility and operating speed.
- The deviation from the existing traditional layouts is a strict minimum.
- The Arabic characters which do not exist on the traditional typewriter keyboard, or do exist in shifted positions, are allocated to the most accessible keys according to frequency plus ergonomic considerations.

This draft standard still doesn't cover selection upper case when aiming at market segments. It is agreed that all differences that exists now between several projects based on traditional typewriter layout will disappear when a 4 shift keyboard (in every language) becomes available which should allow all Arabic characters including vowels to reside on lower shift(s) while retaining numerics and special characters in traditional positions.

Still an approach based on frequency analysis plus ergonomy is needed whether we move toward the new technology of 4 shift keyboard or we remain within the environment of lower and upper shift. We should also mention that unsufficient studies have covered this scientific and global approach. Therefore we may conclude that the keyboard draft standard No. 663 which is supposed to be adopted in October 1985, will be considered as a compromise solution which satisfies most of user /market requirements.

6. 8-BIT ARABIC CODE TABLE

A draft standard has been prepared (No. 662), and is expected to be adopted by ASMO General Assembly before the end of 1985. (5)

This draft standard specifies the properties of a coded character set using 8-bit binary codes for information interchange among different types of data processing equipments using the Arabic language.

This draft standard is based on the following criteria:

- Independancy of the Arabic code in order to reach a final goal which is

Table (1): 8-bit Arabic Code

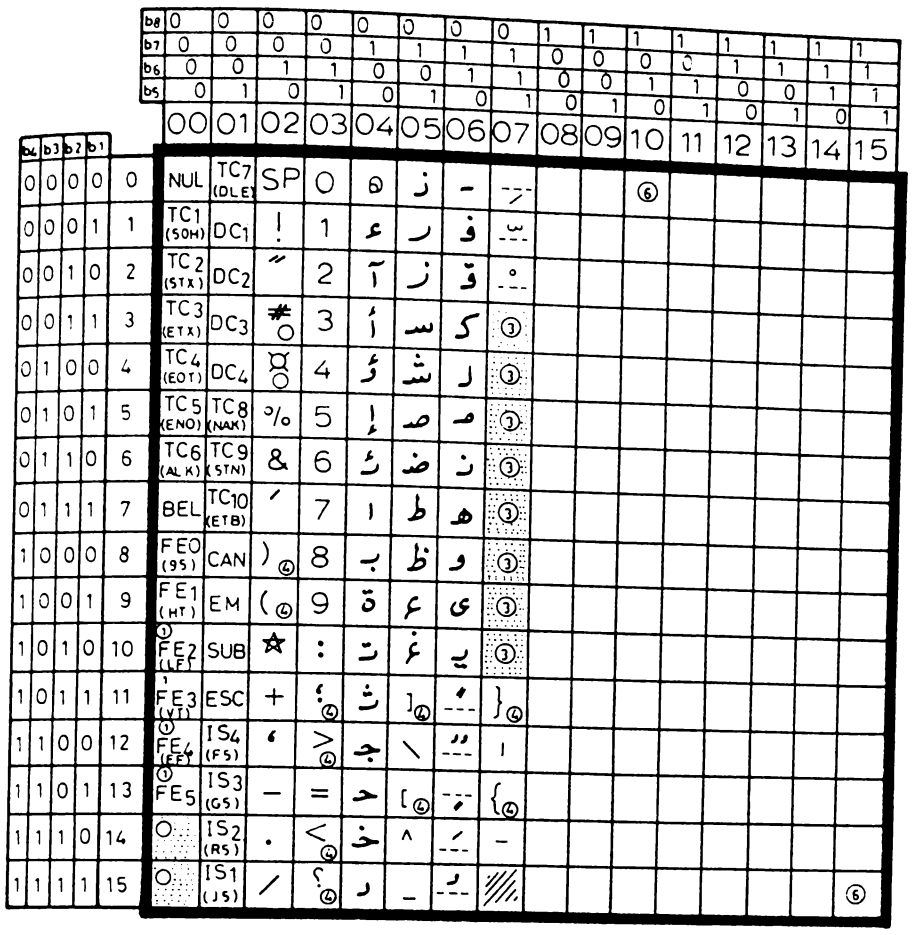
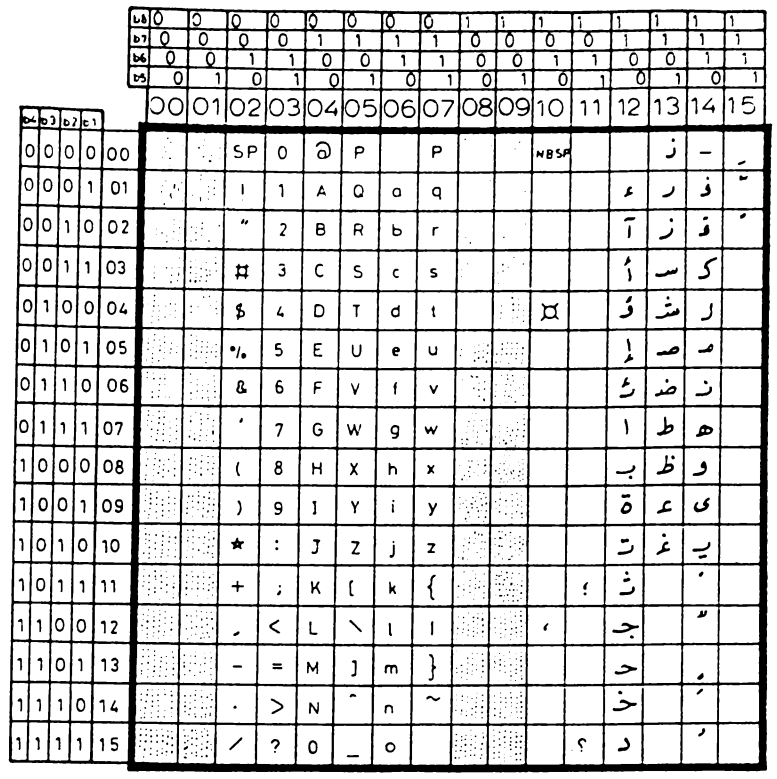


Table (2): 8-bit Arabic/Latin Code



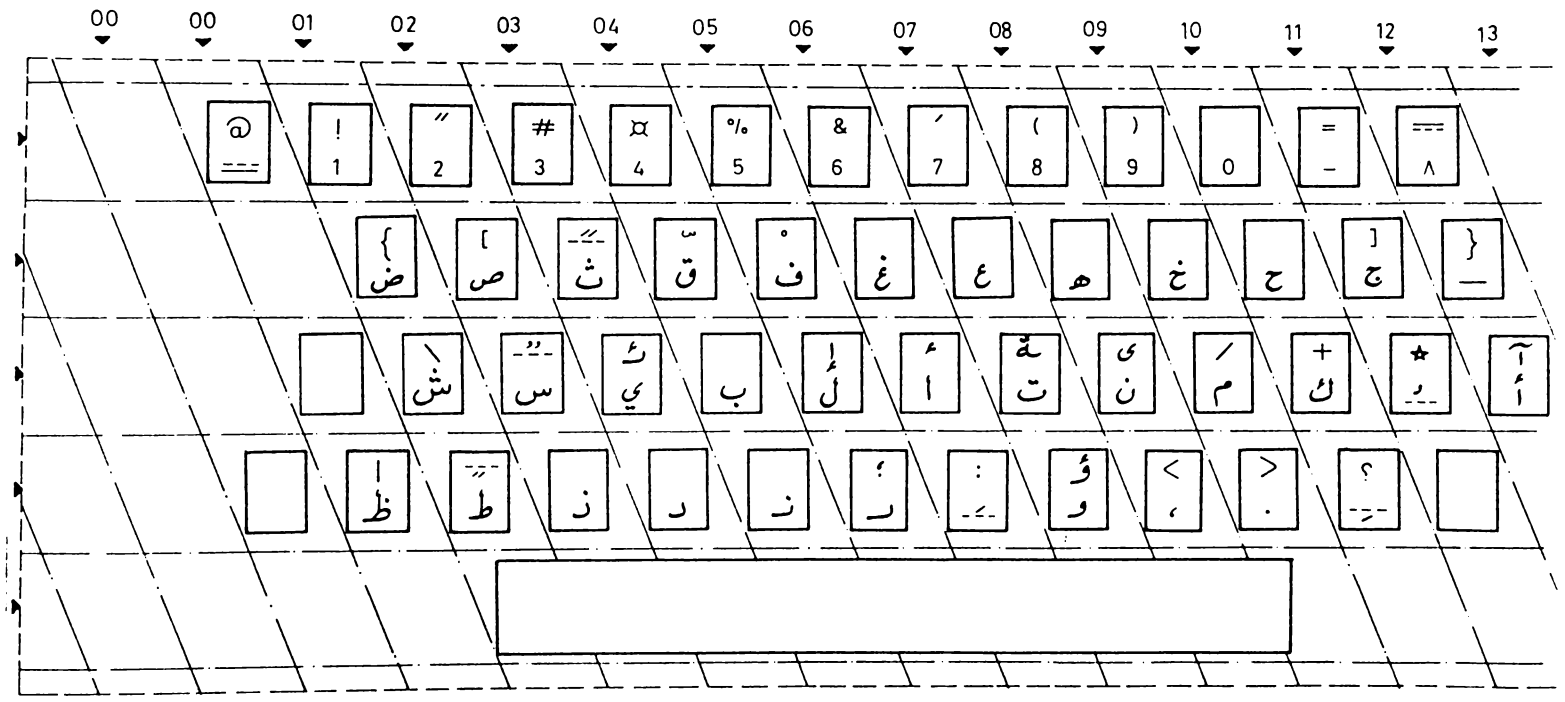


Figure (1): Bilingual Keyboard

The work on 7- and 8-bit Arabic codes will continue. ASMO will compile all relevant comments and problems which might be encountered when these standards are implemented in order to revise them.

Other problems on the agenda also include:

- Arabic bibliographic exchange using magnetic tapes.
- Arabic programming languages.
- Shapes and dimensions of character set to be displayed (on screen, printer, ... etc.).
- Shapes of Arabic characters for optical and magnetic recognition.
- Basic requirements for Arabic Data banks development.

In addition to the above problems, studies must be carried out concerning the specification of peripherals handling mixed data and their communication protocols.

These studies can cover:

- Transmission of mixed Arabic and Latin characters.
- Specifications for Arabized services (Telex, Teletex, videotex, Acess to data books, electronic mailing, ... etc.).
- Data transmission via ARABSAT.
- Coordination as regards networks with other Arab states and the Arab Telecommunication Union.

REFERENCES

1. ASMO 449, "Data Processing 7-Bit Coded Arabic Character Set For Information Interchange" , Oct. 1982.

2. ASMO 445, "Bilingual Teleprinter-Arabic/Latin", Oct. 1982.

3. ASMO 584 (Draft Standard), "Conversion between the 7-Bit Coded Arabic Character Set (ASMO 449) and the 5-Bit Arabic (ASMO 445)", 1984.

4. ASMO 663 (Draft Standard), "Bilingual Keyboard", 1985.

5. ASMO 662 (Draft Standard), "Data Processing-8-Bit Coded Arabic/Latin Character Set For Information Interchange", 1985.

6. ASMO (First Draft Standard), "Data Processing-8-Bit Coded Arabic/Latin Character Set For Information Interchange", 1985.

7. Meeting Report of ASMO TC 8, Damascus, 14 - 16 May, 1985.

8. "Technical Specification For Bilingual Terminal Arabic-Latin", Technical Report Presented at Meeting of ASMO TC 8 and ATU Data Transmission Committee, Damascus, 19 - 20 September, 1984.

The Arabic Coding Standard: Afterthoughts and Desiderata

PIERRE A. MACKAY
Department of Near Eastern Languages
University of Washington
Seattle, Washington, USA

Introduction

The general acceptance of a standard code for information interchange in Arabic has reduced almost all of the studies that preceded the adoption of this code to a purely antiquarian interest. The only point in reviewing older proposals and suggestions is to derive from them some sense of the questions and problems that are not addressed in this standard, and to see whether those questions may be effectively dealt with in later extensions of the standard. The standard as it presently exists provides manufacturers and software developers with a stable platform on which to build, but there are some restrictions and limitations in this standard, notably the fact that it makes absolutely no provision for any of the Arabic Script languages other than Arabic itself. It is far from clear what agency will ever undertake the internationalization of an extended Arabic Script coded character set, and there is a strong probablity that national interests will ensure that the Persian, the Urdu, the eastern Turkic, and the Malay versions of Arabic Script will be encoded in ways that are thoroughly and intentionally incompatible with one another. What follows are some suggestions for an approach which might avoid that unfortunate development

History

In the past decade there has been a large assortment of coding schemes proposed for the Arabic Script languages, most of them unofficial. A few have briefly made the distinction of being adopted as standards or quasi-standards, and the coding system known as ASMO 449 now has full general authority. It might therefore seem otiose to go into the question any further, and doubly so to go into it when one is outside the region most intimately affected by the need for a consistent standard for information interchange. Nevertheless, on a purely abstract level, I believe there is still something to be learned by taking a fresh look at the problem, and by recognizing that it has certain peculiarities which distinguish it from the problems so ably and thoroughly discussed in Charles E. Mackenzie's book, *Coded Character Sets, History and Development*, which reviews the history of the American Standard Code for Information Interchange, and of the very closely related ISO 646 7-bit coded character set.

I propose to look at the problem of Arabic under the two criteria of *Sufficiency* and *Efficiency*. A *sufficient* character set is defined as one which includes all graphic symbols needed for a particular range of computing and communications services. An *efficient* character set is one in which sufficiency is attained in the smallest possible range of values. In these definitions *sufficiency* clearly has priority. Let us look at an application of these criteria to the ISO 7-bit coded character set.

In the current range of applications for the ISO 646 character set, there are two obvious layers of sufficiency. The lower layer is in use in almost every general operating system and file nameing system except UNIX. Not only is the uppercase Latin alphabet sufficient for these applications, it is in most instances enforced by the operating system, which imposes an automatic operation of folding on the user's keyboard input to insure that only the uppercase values are presented to the system. To a very great extent, this level of coding

is a historical fossil left over from punched cards and 6-bit codes, but it is deeply embedded in in the software environment throughout the world, and is likely to remain so even if UNIX continues to grow as the operating system of choice for new computing machinery.

ASCII is not strictly efficient for this range of applications, since it imposes extra computing effort on the I/O system, and since it sometimes puzzles the naive user. But ASCII aims at sufficiency for a second layer of applications, in which the Latin-letter distinction between upper and lower case is of great importance. For these applications, ASCII meets the test of both efficiency and sufficiency, and furthermore, it is designed so that the folding operation to meet the needs of the lower level of applications is as simple as it can be. Mackenzie's study of the genesis of ASCII reveals a deep concern with the necessity of providing for these two layers of sufficiency, although the specific terms I have adopted are not used.

In the history of the various Arabic Script codes, we can see the effect of these criteria either consciously or unconciously applied. Perhaps the most interesting case is Baghdad NCC code which, like so many early North American codes was designed with the restrictions of existing punched card devices very clearly in mind. Baghdad NCC code was extremely restrictive, it allowed no graphic variants in letter forms, no *harekāt* and not even *tashdīd* was provided for. There was obviously a great number of applications that could not be considered with this code. But it was very successful within the range of its intended applications, and like uppercase only Latin-letter character sets, its successes need to be remembered as well as its limitations. For the specific purposes for which it was designed, this code met the test of sufficiency.

The various codes which have appeared under the general name of CODAR are a more complicated case. Unlike Baghdad NCC code, CODAR arose out of a technology rather separate from computation. The great early success of CODAR centered on Dr. Ahmad Lahkdar-Ghazal's striking achievement in creating a set of Linotype matrices which met the test of both sufficiency and efficiency for his proposed Arabe Standard Voyellé **(ASV)**. One cannot review the early publications of Dr. Lakhdar-Ghazal without being deeply impressed by the effectiveness of his system. The character set is striking and idiosyncratic—Maghrebi scripts have always been distinctive—but it is extremely clear, and will stand up triumphantly against many other attempts at modern semi-automated character coding.

The basic assumptions of CODAR were so perfectly matched to the original technology for which it was developed that they carried over into the attempt to adapt the scheme to computation. Here they continued to meet the demands of sufficiency (so long as only Arabic language text, rather than Arabic Script text was considered) but they did not meet the test of efficiency, and by being inefficient, they crowded out certain other potential ranges of application which might otherwise have been possible in a 7-bit code. In essence, they were frozen at the level of technology represented by a pre-formed character printing system, and took too little account of the capacities of local micro-processor systems to produce a wide range of character shapes from a limited repertory of input codes.

By 1978, a compromise coding table had been developed which consisted, in effect, of Baghdad NCC code with extensions from a subset of the CODAR linotype matrices. The resultant code met the test of sufficiency at several levels, but the levels were not really defined, at least not with reference to computerized applications. In a report deriving from a conference held in collaboration with Unesco, dated 22-23 June, 1978, reference was made to a layered arrangement of subsystems derived from ASV-CODAR, but a reference to the diagram of that arrangement makes it clear that the layering had reference to graphic lettershapes rather than to logical elements. Those shapes were directly and exactly derived from the linotype matrices mentioned previously. The effect was to halt on one technology just when its successor made a better available.

The CODAR compromise had a period of great success. By the time of the conference held under the sponsorship of the Saudi-Arabian Standards Organization in Riyadh, in June 1980, a form of CODAR known as CODAR-U had gained such wide acceptance that it was the only scheme given any serious consideration. A proposal by the host organization itself, one which had some very important features that will some day need to be taken into account was so completely disregarded that it has never been published. CODAR-U seemed triumphant by 1981, but it had certain logical weaknesses-above all in that it did not in any way meet the test of efficiency.

ءابتةثجحخدذررسشص
ضطظعغفقكلمنهويى set Co

ً ٌ ٍ َ ُ ِ ّ ْ ٓ ~ set Vo

پچژگڤ(ۋ) set Fo

The minimum alphabet for an Arabic script data interchange coding table

as outlined by Dr. Abdalilah Dewachi, during the Symposium held by the the Saudi Arabian Standards Organization, June 1-4, 1980

Note: The letter forms shown are for reference only and do not imply any decision as to style and appearance of font.

1. Set Co comprises the absolute minimum set of consonantal values of Arabic in the order presented by the CODAR-U coding table.

2. Set Vo comprises the vowels and other diacritics (shadda, madda) required for the production of vowelled text in Arabic.

3. Set Fo comprises the minimum number of consonants from the Extended Arabic Script alphabet needed to accomodate frequently occurring proper names and technical terms: Pe, Chim, Zha, Gaf and Fa muthallath, representing (Pe) an unvoiced Ba, (Chim) an unvoiced Jim, (Zha) a voiced Shin, (Gaf) a voiced Kaf and (Fa--or occasionally Waw--muthallath) a voiced Fa.

1) It included the morphological sign *hā' al-ta'nīṯ* in the the basic alphabetic sequence. This violates the only generally used Arabic lexical sequence without providing any compensatory advantages. Whatever the historical origin of this sign in ancient Semitic, it has, in Arabic, become a subalphabetic sign. Consider, for instance, the fact that it can appear only on nouns and adjectives and only in final position. No normal part of the alphabet is subject to this restriction. The restoration of the forms of alphabetic *tā'*, when the sign happens to to be protected by a suffix, is good evidence of the original historical derivation from the letter, but no normal lexicon of Arabic would give it the weight of an element of the alphabet in establishing a lexical order.

2) By doing so, it made it impossible to confine the basic Arabic Script alphabet to a conveniently defined set of bit-patterns. This weakness is in fact shared by ASMO 449. It appears to be a price which is considered worth paying for other reasons, but it is a price, and it is an impediment to the establishment of a hierarchy of levels of sufficiency and efficiency.

3) It filled up columns *4* and *5* of the coding table with variant forms of the basic alphabet and of the signs of vowelling. These variants are otiose in a properly designed software environment, and inadequate in the absence of software for context evaluation.

4) It made a gesture in the direction of the letters in the extended Arabic Script alphabet, but failed to provide even the full set of extensions as used in Persian. Persian *peh* and *gāf* were supplied, but *çīm* and *ẓā* were not. Instead, a *fā' muṭallaṭ* was provided which, in the absence of *çīm*, is a little like including a character for *'ayn* in Latin-letter ASCII.

For various reasons, most of which I am unaware of, CODAR-U had a very short life as a quasi-standard, and has been replaced by an entirely different mapping of characters in ASMO 449. ASMO 449 improves very greatly on CODAR-U for efficiency, but it continues to offer a basic consonantal character set which extends over a poorly defined range of bit patterns, and I learned already on my first visit to Damascus in April how annoying that can be when it results in the scrambling of a filename on a system which assumes that the characters of columns *6* and *7* have the same lexical value as the corresponding characters in columns *4* and *5*. A cure for this problem would be for everybody to switch at once to the UNIX operating system, but since that is unlikely, and since distributed processing on heterogeneous networks, which is coming faster than we can prepare adequately for it, is likely to keep the only non-UNIX filename restrictions around for quite a while, I propose to explore the ways in which a coding scheme might be built up through layers of increasing logical functionality. In one sense, this may be wasted effort, but in another sense, any new logical way to look at a problem in computing has the possibility of being productive.

Building a Layered Character Set.

This exercise derives from an exchange of letters with M. Robert Gabriël of the Royal Library at Brussels, who was associated at the time with ISO/TC46 for documentation standards, and also from discussions with Dr. Abdalillah Dewachi, who suggested the following breakdown of the Arabic Script character set in Riyadh, in 1980.

Dr. Dewachi's subsets point in the right direction, but still leave us with an oversized basic character set. If we really look at the absolute minimum for communication with an operating system, and for file names, we can get the number down to 28 consonants. No considerations of graphic shape are involved in this selection, and the 28-character set cannot be used to write correct Arabic. It can be considered as *sufficient*, however for the restricted applications of communication with an operating system, naming of files, and programing languages. There is no *hamz^{ah}*, final *hā'* is used for *hā' al-ta'nīṯ*, and it is assumed that character generation software will provide dots for initial and medial *yā'* but leave them off final *yā'*, so that it can do double duty for *alif maqṣūr^{ah}*. It is impossible to write Arabic correctly with such a character set, but for a range of applications limited to nouns (filenames), simple imperative and conditional statements (programing languages), and simple declarative statements (comments), such a character set would be sufficient, and it would offer one outstanding advantage in efficiency—it would fit in columns *4* and *5* of a 7-bit table.

In suggesting columns *4* and *5*, I am breaking from the organization proposed in the correspondence with M. Gabriël. Four years ago it still appeared that the attempt to retain a degree of consistency with the placement of consonantal codes in columns *6* and *7* as they appeared in both Baghdad NCC code and CODAR-U was desirable. Now that that constraint has been abandoned, there are very good reasons to move entirely into columns *4* and *5*. Where any folding operation takes place in a standard operating system it tends to be the total folding of columns *6* and *7* into columns *4* and *5*, so that if there is to be a minimum sufficient character set, it should be in columns *4* and *5*.

Here is a table showing the minimum set of codes in Arabic Script.

ا	ظ		
ب	ع		
ت	غ		
ث	ف		
ج	ق		
ح	ك		
خ	ل		
د	م		
ذ	ن		
ر	ه		
ز	و		
س	]		}
ش	\		|
ص	[		{
ض	'		~
ط	ي		

On the basis of similarity of function the next larger layer is chosen to include all those characters which can successfully be folded into the minimum set. The resultant set meets the test of sufficiency for Persian as well as Arabic, and it can also be claimed as sufficient for continuous Arabic text up to a certain level of complexity. The absence of the signs for *taṣdīd* and *hamz^{ah}* is the most serious omission here, but it is unavoidable if the criterion of folding is to be maintained. *Hamz^{ah}* might appear to be acceptable in the position which

would fold to *alif* but then the results of folding an interior *hamz*^{ah} over a *yā' kursī* would be unsightly and misleading. Note, however, that filenames, etc. may be typed into the data entry station using this larger subset and even if the operating system folds the extended characters back into the minimum set, the result will remain readable.

Table two shows the extended set of consonantal codes in Arabic Script.

ا	ظ		
ب	ع	پ	
ت	غ		
ث	ف		ڤ
ج	ق		
ح	ك		گ
خ	ل		
د	م		
ذ	ن		
ر	ه		ة
ز	و	ژ	
س	]		}
ش	\		ا
ص	[		{
ض	-		ـ
ط	ي		

Now it is possible to create the next layer of sufficiency, which will provide for correctly written Arabic in all senses. There are no really serious constraints on the positioning of the next group of subalphabetic characters in an Arabic coding scheme, but some sort of grouping by function seems reasonable. It is unfortunate, but unavoidable, that in the nature of Arabic morphology, all normal systems of alphabetization, whether by radical in the traditional sense, or by letter sequence in the manner of Persian and other non-Semitic Arabic Script languages require that the vowels be stripped out of the text, or their effect neutralized by programming. Since that is the case, however, there is no clear reason to put them in one position rather than another. Classical prejudice suggests that they be kept in the order that represents their usual presentation as case endings, with *ḍamm*^{ah} first, then *fatḥ*^{ah} then *kesr*^{ah}, and it seems reasonable to have the *nunated* forms closely related to the simple vowels.

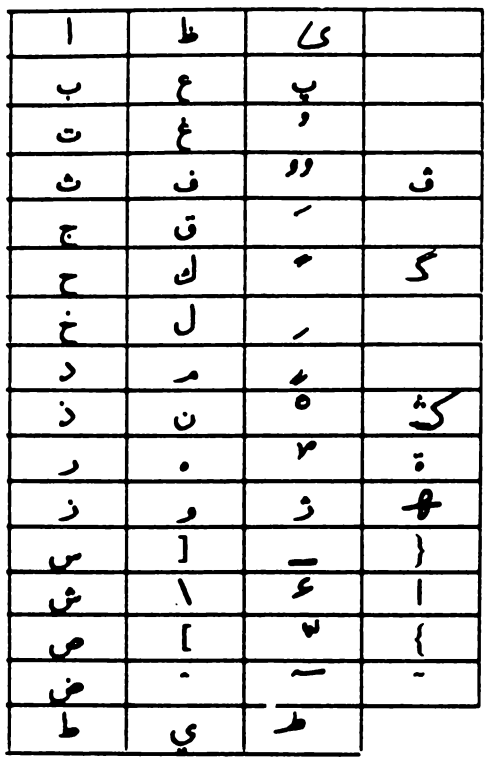
Even more difficult is the handling of *taṣdīd*, which requires the substitution of the preceding radical letter before a correct alphabetization can be attempted. (It may be noted here that English historical orthography has been blamed for a great many things, but it confers one great benefit on the users of the language. The elimination of all diacritical marks has made the task of alphabetic sorting far easier in English than in almost any other major language except Russian. The effort in Czech, for instance, must be much greater. The proper handling of Arabic doubled consonants will be similar.)

Table three shows the extended set of consonants together with the vowels.

ا	ظ	ى	
ب	ع	پ	
ت	غ	و	
ث	ف	ٷ	ڤ
ج	ق	ـَ	
ح	ك	ـٙ	گ
خ	ل	ـُ	
د	م	ـٗ	
ذ	ن	ـ٘	
ر	ه		ة
ز	و	ژ	
س	]		}
ش	\	ـٔ	ا
ص	[	ـٕ	{
ض	-	ـٓ	ـ
ط	ي		

When any characters from this new layer are included in a file, folding ceases to be acceptable, since the results will be incomprehensible. On the other hand, this character set is the minimum sufficient for work on speech recognition and speech synthesis. It has certain other advantages as well, which have often escaped notice. When I worked out the **KATIB** typesetting system in 1975, I had no access to CRT input devices, and was forced to develop a punched card coding system in romanized form. To make this even minimally intelligible for proofreading purposes, it was obvious that the vowels had to be represented, and the input file for the one book produced with the **KATIB** system was written out, more or less accurately, with full *i'rāb*, case endings and all. The presence of all the appropriate *harekāt* was sufficient to allow the program to generate the correct *kursī* for *hamz*^{ah}, and even to provide for forcing an incorrect choice where it was necessary to retain a manuscript variant. One of the special problems that has repeatedly surfaced in the development of coding tables of Arabic has been the positioning of *hamz*^{ah}—it accounted for the only redundancies in the proposal of the working committee of SASO in 1980. But the use of the obvious vowels before and after *hamz*^{ah} provides an unambiguous key, based on normal educated pronunciation, which will result in the

correct *kursī* in every case. It is, of course necessary to include software which will choose a form of *yā'* without the two dots, but this will be increasingly easy, particularly in systems where the display is driven by an independent processor. For hard-copy printing systems there is no problem at all.



The final *yā'* without *niqāṭ* which serves for many instances of *alif maqṣūr ah* is rather more of a difficulty. Full historical vowelling would usually result in the correct character, but since the whole point of the term "*alif maqṣūr ah*" is to indicate that no one pronounces the full vowelling it would be very difficult to make this work. The adoption of the Egyptian typefont style would get around the problem very neatly, but it since this has not become general in the past it is not likely to in the future. It would appear that a separate character ought to be provided. Two other characters would probably be included among the desiderata of most users, one to create an on-line join, and the other, which I have never seen in an officially published coding system, but which I have often had occasion to advocate to a sympathetic audience. This is an affix separator—a character which could print or not in more or less the same way that included *harekāt* will sometimes be displayed and sometimes not. For computing tasks involving lexical sorting, a clear marker between *alif-lām* and the following noun would be very useful. Otherwise it will be extremely difficult for the software to maintain the distinction between *albāb* and *al-bāb*. (There is no obvious distinction on the printed page either, but a human reader controls a range of general context which is rarely accessible to the computer.)

This really concludes the exercise, and I emphasize that it is simply that, an exercise in developing a character set based on layered levels of sufficiency for various computing tasks. Within the framework of a Local Area Network, it might be possible to evaluate the usefulness of such an approach, particularly in a network which included a number of heterogeneous processors and marginally incompatible operating systems. There are a number of spaces still open in the coding table, and some of these might be pressed into service to provide for the Malay, Pashto, Turkish, Urdu or other similar character sets. The floating *ṭā'*, for instance, and the two-eyed *dō-çeşme he* would be obvious candidates for inclusion. I confess to a strong bias in favor of a character set that includes a *gnaf*. But these are all simply refinements on the principal idea, which is to explore the possibility of putting together a character set based on layered subsets offering different levels of sufficiency for different tasks.

References

For a general bibliography of the official references to the development of the Arabic Coding Standard, see the paper "Standards for Arabic Characters in Informatics", page 106, above.

Mackenzie, Charles E. *Coded Character Sets, History and Development*. (Addison Wesley, Reading, Mass. 1980)

Arabic Character Recognition

F. HAJ-HASSAN
Scientific Studies and Research Center
P.O. Box 4470
Damascus, Syria

1. INTRODUCTION

The earliest research on character recognition started more than 20 years ago. The problem of recognition is generally composed of 3 parts (figure 1) : segmentation, description of form by a set of characteristics or measures and classification.

Printed Latin characters have no problem in the first stage, as they are already separated by spaces. That is not the case for Arabic characters. A word in the Arabic language is composed of sub-parts, each one of which is a sequence of connected characters or an isolated one. So segmentation of words and characters in the word, in order to have an isolated character to recognize, is the first problem to resolve.

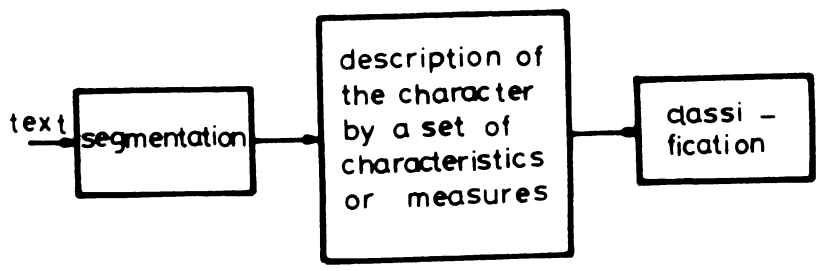


Fig. - 1 - Character recognition stages

Once a form (character) has been isolated, the many different techniques used to solve recognition problems may be grouped into two general approaches : the statistical or decision-theoretic approach and the syntactic or structural approach;

In the statistical approach, the form is characterized by a set of measures x_i called "features". Each pattern is represented by a feature vector $\vec{x}$. Its recognition is usually made by partitioning the feature space into sub-spaces. Each sub-space corresponds to a pattern class. The choice of the set of features and decision-rules for partitioning are very important in this approach. That is not easy even for a pattern space composed of a few classes, and it is very difficult for pattern composed of many classes such as characters.

The syntactic approach is applicable for patterns composed of a set of recognizable sub-patterns called "primitives". The recognition of each pattern is usually made by parsing the pattern structure according to a given set of syntax rules.

This approach is applicable for the recognition of characters which can be described by a set of primitives and their composition relations. But two problems have to be resolved:

1) Optimum choice of primitives, so that two forms from two different classes can be distinguished, even if these forms have very near structures such as: (fa ف , ghin غ). That means the use of a very fine description which is too much fine for other characters. A very complex inference grammar may be the result. We suggest recognition in steps (hierarchical recognition) to resolve this problem.

2) In character recognition, there are many classes. It is very hard to infer a grammar Gi for the class i, taking care at the same time of the set of description lj ($j \neq i$) of other classes. The grammar Gi is inferred so that $li \in L(Gi)$; where li is a set of sentences describing the set of training samples of the class i, L is the language "set of sentences" generated by the grammar Gi. A priori verification is made by ensuring that the languages Li and Lj are disjoint. If they are not disjoint, Gi and Gj have to

be changed, and so on ... That means we deal with the problem of iteration, and must ensure convergence to a solution.

A hierarchical recognition procedure resolves this problem also. Since ambiguous cases are allowed in the first step of recognition, grammars generating disjoint languages are not necessarily inferred in this level. In the second level of recognition, dealing with classes with restricted populations, specific grammars are inferred which generate disjoint languages.

We will briefly review some syntactic models used in character recognition in the section or primitive extraction.

Our object is to read text by a TV camera and recognize its characters. The system is simulated on a VAX 11/780 computer, to which the camera is connected. The text is presented by a binary array where points belonging to characters take the value 1 and the others take the value 0. In this lecture we illustrate our segmentation and primitive extraction procedures.

2. SEGMENTATION

The segmentation of lines in Arabic printed text is achieved simply by detecting spaces between them on their projection on a vertical axis (fig. 2)

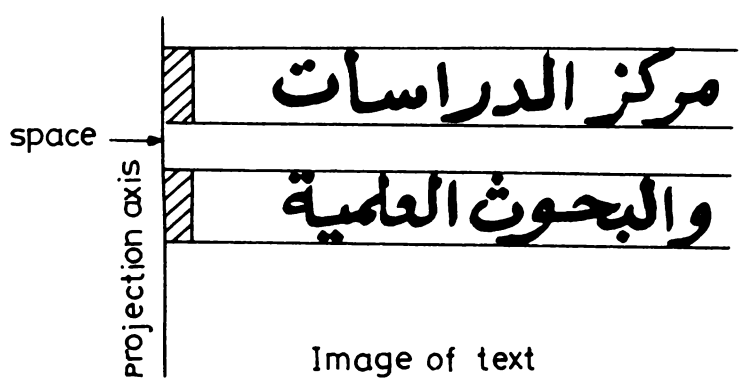


Fig. - 2 - Segmentation of lines.

As we mentioned above, a word in the Arabic language is composed of subparts. Segmentation of words needs to distinguish the interword space from spaces between subparts in the word. In the training phase a minimal space between 2 words is defined. It is always bigger than the maximum space between subparts.

To allow some tolerance in the size of characters, we define all measures relatively to the height of the line.

Segmentation of characters in the word requires the study of their forms and connection rules. We give here some of their properties:

* The length of an Arabic character is variable. It depends on its position in the subpart. It is maximal either when the character is situated at the end of the subpart or when it is an isolated character.
* The characters generaly begin with a "distinctive shape". So there is always a "thick part" at the begining of the projection of a character on a horizontal axis.
* An Arabic character is connected from the left side to the following character by a "horizontal segment". So there is a "thin part" at the end of a connected character. Detection of this "thin part" is a necessary condition to define the end of a connected character but it is not sufficient. Some characters like (sin-" ـ ") have "thin parts" in their middle also. Moreover, many characters have "points" above, inside or under them.
* Many characters at the end of a subpart, whether connected from the right side or isolated, do not have "thin parts" at their ends.

We suggest using two criteria for segmentation : thickness and length.

An unsupervised training phase shows that:

- The length of subparts composed of two characters only is always more than the maximal length of isolated characters.
- The "minimal length" of a connected character and the "maximal length" of an isolated character or a character at the end of a subpart can be defined.

The procedure succeed in separating all training characters(100 words, each one composed of 5 characters on average) except in two following special cases:

. The negation particle " لا ", composed of two characters: (lam - " ل ") and (alef - " ا "), is taken for one character.

. The definition particle " ال " composed from the same two characters; but in the inverse order, is taken some times for one character (fig. 3 - a).

These cases were expected because of the characters of which they are composed and the special form of the negation particle. This particle is even treated by typewriter as an isolated.character. We suggest resolving this problem in the stage of classification.

The segmentation procedure is also successfuly tested on other font (Triumph). Some results of

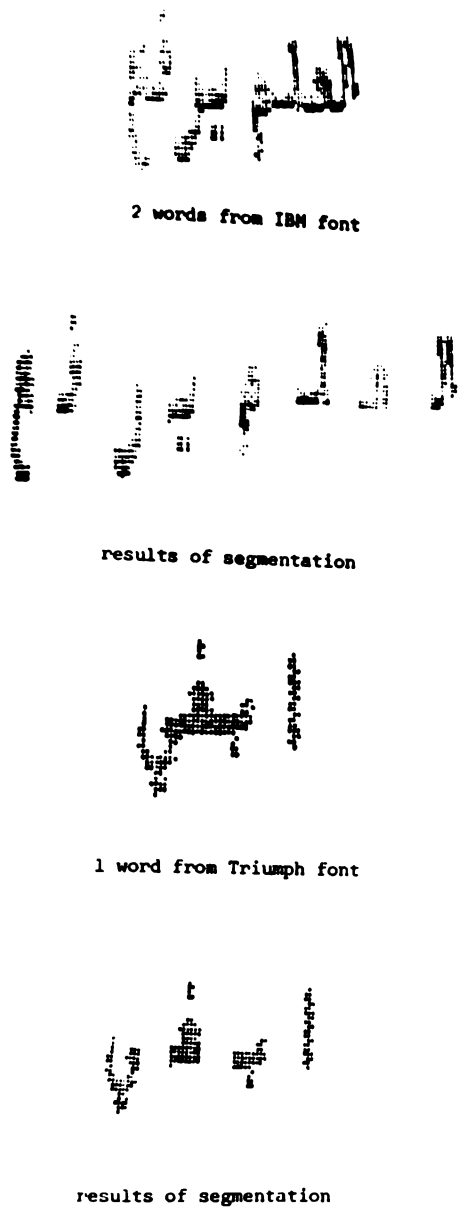


Fig. - 3 - Examples of segmentation

segmentation are given in fig. 3 .

The suggested procedure gives, beside segmentation of characters, the following information useful in the stage of recognition:

* height of line in the text
* position of character in the word: first , second ...
* situation of character in the word: connected from the left side or not
* length of character

3. PRIMITIVE EXTRACTION

In 1969 Shaw [1] has proposed a language for Latin characters. He considers four types of primitives: verticals, horizontals, obliques with positive slope and obliques with negative slope. The association rules adopted here are those used in PDL language (Picture Description Language [1]). This language leads to a context-free grammar. Recognition needs a complex syntactic analysis of the shape description.

Narasimhan proposes [2] also a context-free grammar for generation of Latin characters. Each character is written in a fixed rectangle. Seven regions are determined in this rectangle and a set of primitives (verticals, horizontals, obliques and curb element) is used with a set of attributes. Each primitive has as its attribute some of the regions . The use of regions simplifies the description of characters, as the concatenation rules are implicity considered by reference to the regions.

Chehikian uses [3] the four types of primitives used by Shaw and the idea of predefined regions of Narasimhan. He defines 29 binary variables describing the existence of primitives in the regions.

We have proposed [4, 5] an analogue procedure with a refinement in the position of regions and hierarchical recognition for Latin multifont characters.

For Arabic characters, we propose using the idea of regions also. That permits a description of the form with a binary word which is easy to deal with and to implement. The extraction algorithm of the four previous types of primitives acts inside the area limited by the rectangle situated just around the structure of the character (fig. 4). So points above or below the character have to be eliminated . An elimination procedure can be used to detect the existence and the position of points situated above or under the character. Points inside (fig. 4 - b) will be detected in the step of recognition.

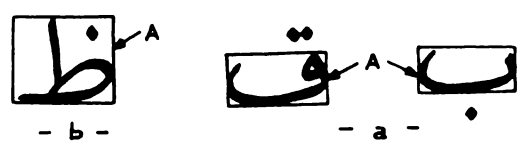


Fig. - 4 - Working zone of primitive extraction algorithm

3.1 Point Detection

Detection of points in a character is simply made by projection on a vertical axis and detection of "space" (fig. 5). If there is a point, comparision between the length of the 2 distinct parts of

projection, permits a definition of point position: above or below the character.

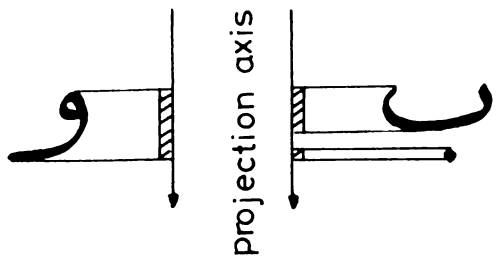


Fig - 5 - Detection of points

This primitive together with the segmentation procedure divides the set of Arabic characters into 6 groups (fig. 6).

3.2 Extraction Algorithm For Principle Primitives

The extraction procedure is based on environment analysis of each point of the shape. Four environmental characteristics attributes are calculated : IV, IH, G, D. They define the type of primitive to which the point belongs.

Let us take the point M for example (fig. 7). We call :

the distance H1 H2 by IH
the distance V1 V2 by IV

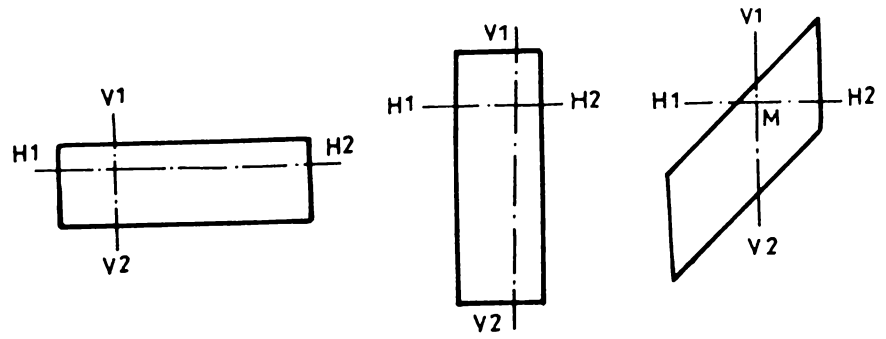


Fig. - 7 - Attribute IV(V1, V2) and IH(H1, H2)

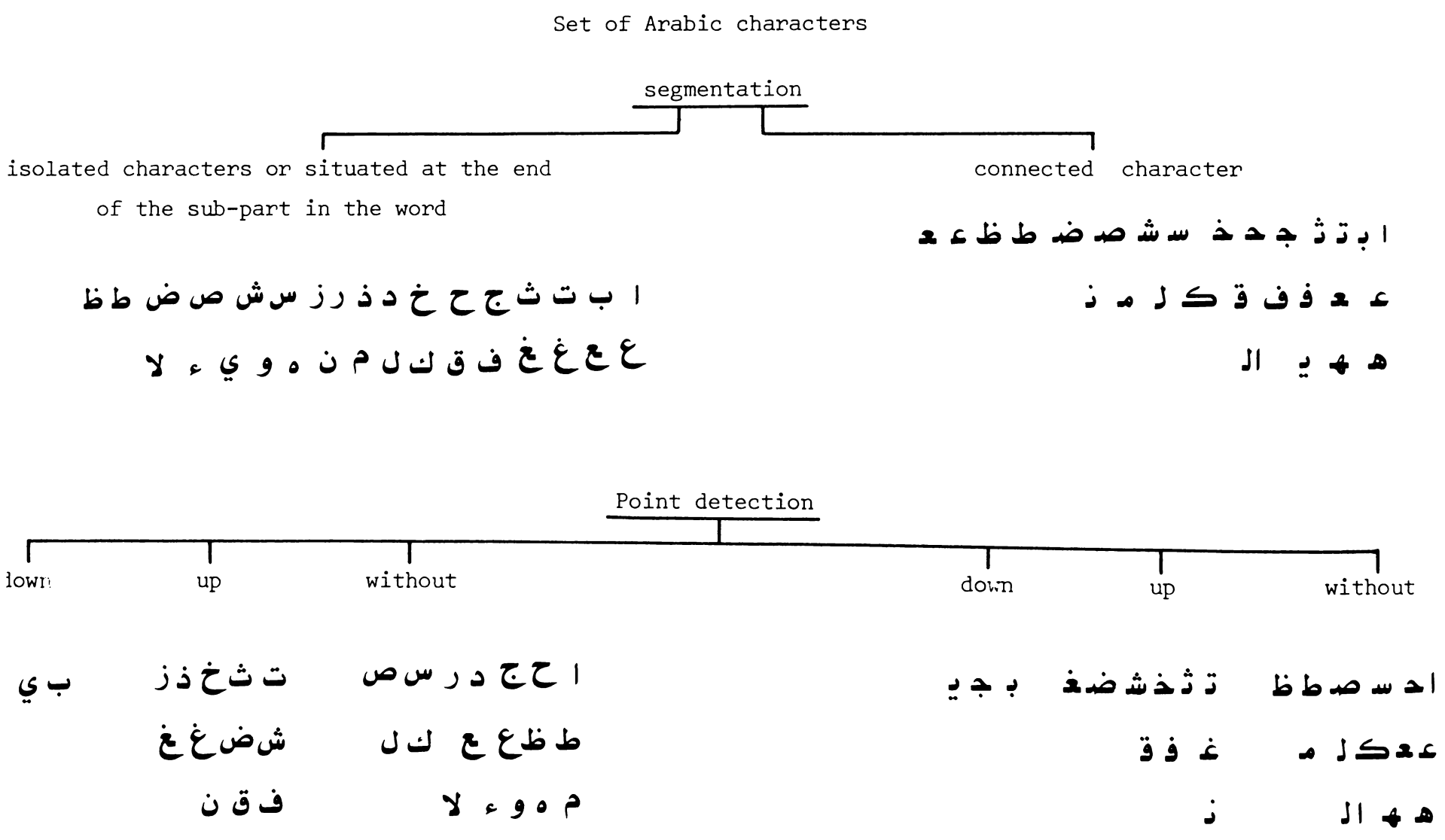


Fig. - 6 - Groups resulting from segmentation and point detection.

In the rectangle a b c d defined by H1 H2 and V1 V2 (fig. 8), we consider the pair of points (m1, m3) and (m2, m4) situated in the first, third, second and fourth quadrants and symmetrical relative to M. We call:

D = card (m1, m3)

G = card (m2, m4)

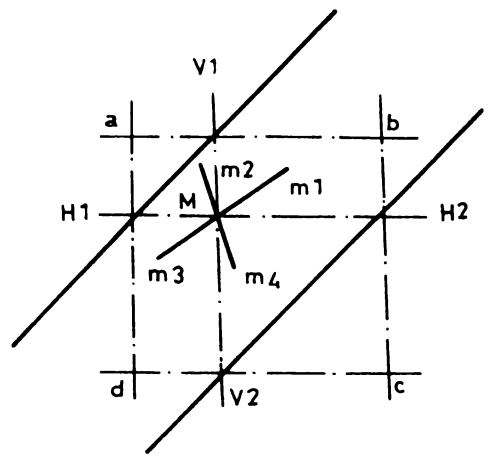


Fig. - 8 - Environment of point M (rectangle a b c d).

Sets of (m1, m3), (m2, m4) in different types of segments: vertical, horizontal and oblique are shown in fig. 9. It shows that:

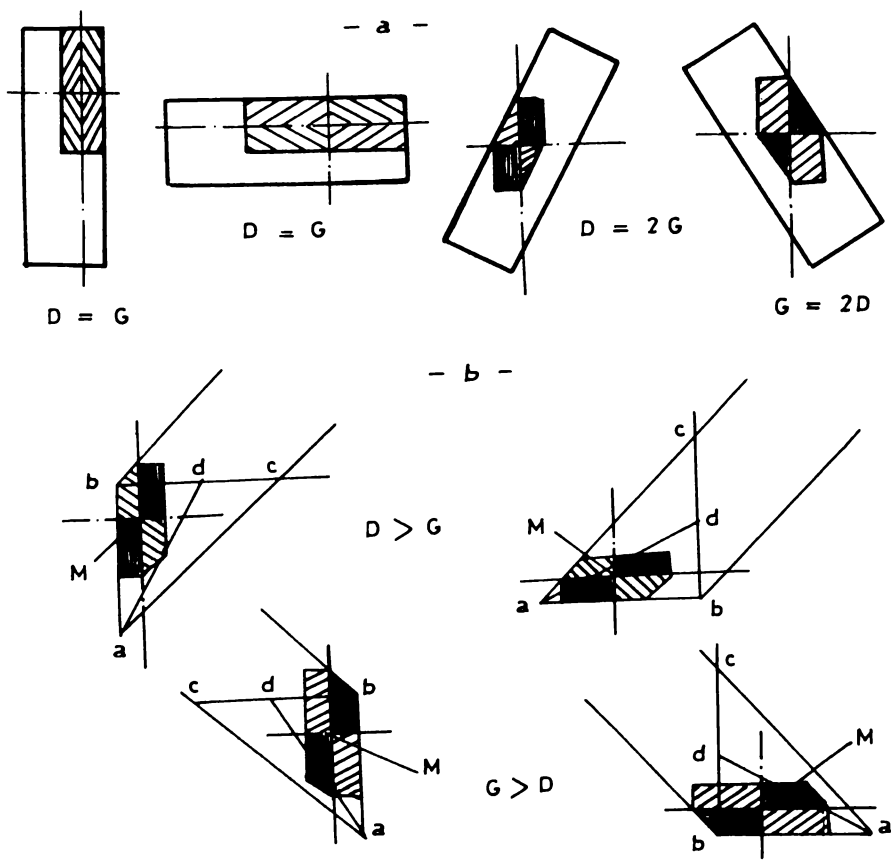


Fig. - 9 - Sets of (m1 , m3) and (m2 , m4)

- if M ∈ vertical segment IV > AV. IM and D = G where AV is the relation between the minimal length and maximal thickness of a vertical segment
- if M ∈ horizontal segment IH > AH. IV and D = G; AH has an analog definition to AV
- if M ∈ oblique segment:
  with positive slope : D > G ,
  with negative slope : G > D
  for all points even for those which are situated in the triangle a b d (fig. 9 - b, a d is the median of a b c).

In the results of a supervised training on Arabic letters from IBM font, following criteria appears:

- if M ∈ vertical
  the criterion IV > 1.5 IH is always realized
  = = D = G is nearly realized. In fact
  = = $0.9 < \frac{D}{G} < 1.1$ is always realized
- if M ∈ horizontal
  the criterion IH > 2.2 . IV is always realized and it is sufficient.
- if M ∈ oblique
  the criteria D > G or G > D are generaly realized. They are not realized for points situated on the borders of the segment. If we consider the attributes $\grave{D}$ and $\grave{G}$, calculated in the same way as D and G, but for an increased surface as it is shown on the fig. 10, the following criteria are always realized:

  $\grave{D} > \grave{G}$ for oblique with positive slope
  $\grave{G} > \grave{D}$ = = = negative slope

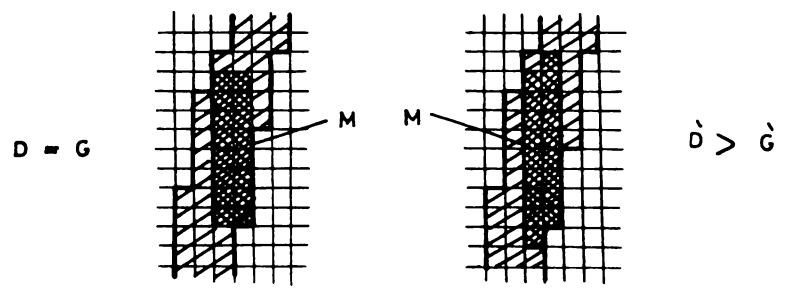


Fig. - 10 - Attribute D, $\grave{D}$, G, $\grave{G}$ for point on the board.

The choice between D, G and $\grave{D}$, $\grave{G}$ affects the classification of the points from vertical and oblique segments. The horizontal primitives are not affected, as the criterion IH > 2.5 IV is sufficient. It is

clear that:

- when D and G are used, vertical primitives appear clearly while appearance of oblique primitives is decreased.
- if $\grave{D}$ and $\grave{G}$ are used, oblique primitives appear clearly and vertical ones are weakened.

To ensure good detection of important vertical segments in the character, we suggest using the attributes D and G for the extraction of vertical primitives, for each point having $IV \rangle \frac{1}{2} N$, where N is the height of the character. This provides good detection of both vertical and oblique primitives.

Creteria chosen sometimes leads to the classification of some points into 2 types of primitives : horizontal and oblique or vertical and oblique. This situation appears especially in case of an oblique with a steep slope relative to the horizontal or vertical axis. Some erratic points appear. They are eliminated by:

- preprocessing the primitives in order to keep only the points neighboring other points in the same direction of the given primitives.
- taking the decision for points belonging to oblique primitives, only when they do not belong also to a vertical or horizontal primitive.

Some characters have long vertical segments. They can be detectd simply by the criterion: $IV > 0.8\ N$. Figure 11 presents an example of primitive extraction

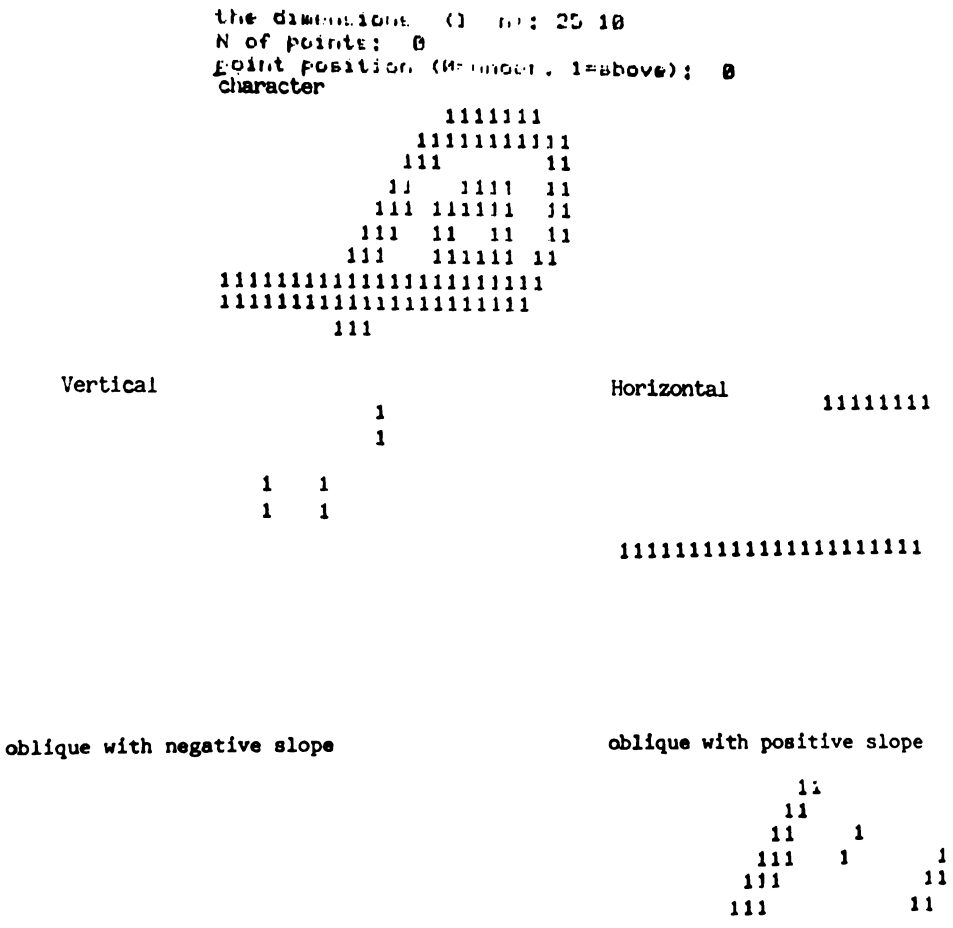


Fig. - 11 - Results of primitive extraction procedure on the character ha (character Ha - " ﻫ ").

It shows that horizontal, vertical and oblique primitives are well detected.

4. CONCLUSION

We suggest a simple and efficient procedure for the segmentation of Arabic characters. We show that hierarchical syntactic procedure is a good approach for characters recognition. Using the "point" primitive, pattern subspace is devided into 6 subspaces. Each character is decomposed by a given algorithm to its vertical, horizontal and oblique components. So the character is ready to be described by a sentence or binary word in order to be recognized. That will be our next step.

REFERENCES

1. Shaw A.C., A formal picture description scheme as a basis for picture processing systems, Information and Control, 14, 9 - 52. 1969.

2. Narasimhan R., On the description, generation and recognition of classes of pictures, in Automatic Interpretation and Classification of Images, el. Grasselli, Academic press, New York 1969.

3. Chehikian A., Conception et realization d'un automate de reconnaissance de caratères alphanumeriques, thèse d'etat E.N.S.E.R. Grenoble 1978.

4. Haj Hassan F., Procedè de reconnaissance structurelle de caractères alphanumeriques multifontes, thèse d'etat E.N.S.E.R. Grenoble 1980.

5. Haj Hassan F., Chehikian A., Procedè de reconnassance structurelle de caractères alphanumeriques multifontes utilisant des niveaux de decision multiples, Annales de telecommunication, Tome 36, No. 7-8, 401-411, 1981.

6. FU K.S., Syntactic models in pattern recognition and application, Pattern Recongition in Practice, Amsterdam, 303-314, 1980.

7. Badie K., Shemura M., Machine recognition of Arabic cursive script, Pattern Recongition in Practice, Amsterdam, 315-323, 1980.

Electronic Speech Synthesis Arabic Computer Speech Output

M. MRAYATI
Scientific Studies and Research Center
P.O. Box 4470
Damascus, Syria

I. INTRODUCTION:

Within the next few years, speech processing systems are expected to affect almost every aspect of our daily lives: household appliances, voice warning, and toys, voice data-entry reservations and credit verification, low cost reading machines for the blind and voice actuated systems ... etc. It is estimated that the world market in speech (recognition only) in 1985 is of the order of $400 million [1] , and in 1990 it will be $ 3217 million.

It is expected that electronic speech recognition will be a technology of the nineties by which time electronic speech synthesis technology should be firmly entrenched [2] .

The Arabic language is spoken by more than 150 million people, and if it is to be present in modern applications in communications, computers, automated offices etc ., a lot of research is needed. It is quite evident that if the interfacing between the processing computer and the Arabic language, both spoken and written, is achieved, we would see better development of management, industry, science and education in the Arab world. This school has moved one step ahead in this important and vital field of scientific research in the Arab world.

We can distinguish two major types of speech processing namely: Electronic Speech Synthesis (ESS) and Electronic Speech Recognition (ESR). Speech synthesis is the production of spoken messages from their parametric transcription.
Automatic speech recognition (ASR) or understanding is the production of a written message corresponding to its spoken version. In order to understand the problems involved in speech processing a brief review of the mechanism of speech production will be given and reference to the Arabic phonetic system will be made.

I.1 The Vocal Apparatus [3] :

The human vocal apparatus is composed of two main parts, figure (1) :

- The vocal source which includes, essentially , the lungs, the trachea and the larynx.
- The vocal and nasal tracts which form the supra-glotal cavities.

The lungs play the role of air reservoir; air is forced from them, it passes through the trachea into the pharynx. The top of the trachea is surmounted by the larynx. The larynx is a cartilaginous frame housing two lips of ligament and muscles which are the vocal cords. The orifice between the cords is called the glottis.

The supra -glottal tracts are composed of three principal parts: the pharynx, the oral cavity and the nasal tract. The pharynx and the oral cavity form the vocal tract. The nasal tract is coupled in parallel with the vocal tract by means of the velum.

The vocal tract changes its volume and form because of the movements of the larynx, tongue, jaw,lips and the muscular walls. For the adult male the length of the vocal tract is, on the average, 17 cm and the length of the nasal tract is 12.5 cm [4] . The cross sectional area of the vocal tract, A(x), varies with distance (x) from the vocal cords to the lips. It also varies with time according to the phoneme produced, namely with the articulation, and consequently we denote it A(x,t). This area changes between zero cm^2 and about 20 cm^2. On the other hand the cross sectional area function of the nasal tract is variable only over the first 3 cm which couple it to the vocal tract. We call the function A(x) the air function.

The vocal apparatus can be represented by a mecano-acoustic system that is shown in figure (2).

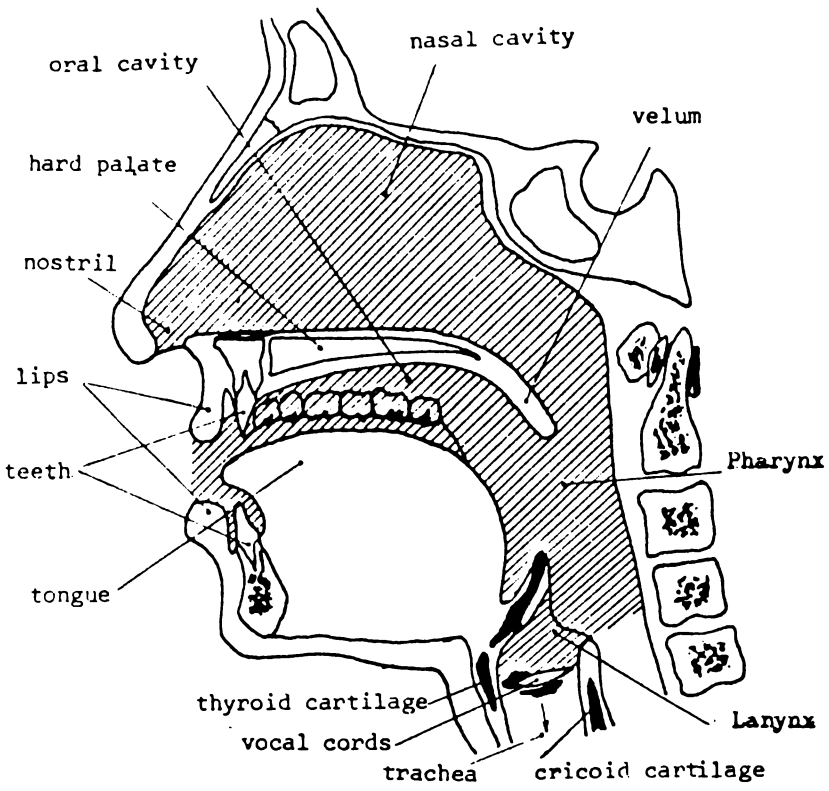


Figure (1): Supra-glottal cavities

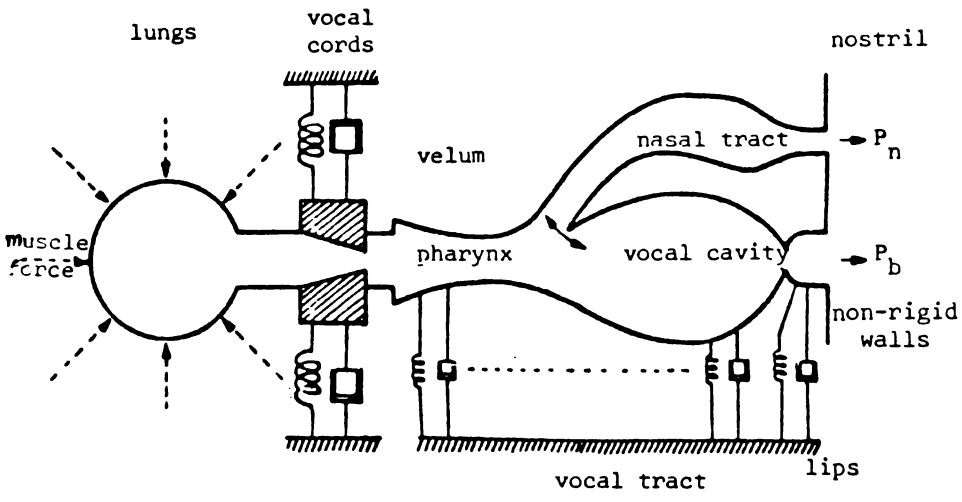


Figure (2): Schematic diagram of the vocal apparatus

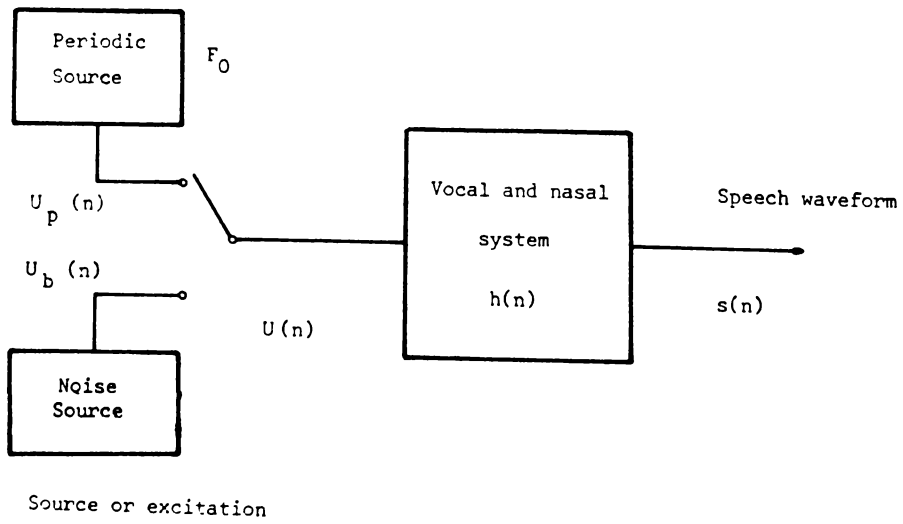


Figure (3)

I.2 Mechanism of Speech Production

The speech waveform s(n) is the response of the vocal system when excited by one or two sources. It is modeled as shown in figure (3) .

The elastic structure of the vocal cords enables them to vibrate rapidly (50 - 500 HZ) and produce, at the output of the larynx, a succession of air pulses . These pulses represent the first source which excites the supra-glottal cavities to produce what is called the voiced sounds. These pulses form an acoustic signal $U_p(n)$ which, before radiation at the lips, propagates through the vocal tract and is spectrally shaped by the transfer function H(w) of this tract . The spectrum of this source signal is a line spectrum, the envelope of which shows an attenuation of - 12 db/octave. It is a periodic signal whose period $T_o = \frac{1}{F_o}$, and F_o is the fundamental frequency or the pitch.

The second source is that of noise, and is localized along the vocal tract, at a point which depends on the pronounced sound. The spectrum of the signal of this source is " flat " at the mid-frequencies and is continuous. It shows an attenuation below 1 KHZ and over 8 KHZ.

The vocal tract can be represented by an acoustic tube with a variable cross section as a function of time, and distance from the larynx A(x,t). The nasal tract is also another tube connected in parallel with the first. The transfer function of these tracts is characterized by several resonance frequencies which pass and amplify the corresponding frequency components of the source signal. The frequencies of the resulting maxima at the output of the tube are called "FORMANTS". The first three of them, F_1 F_2 F_3 , are the most important. By contrast, the minima of the output spectrum produced due to the sections of the tube situated behind the excitation source (fricative sound) or due to the nasal tract are called "ANTI-FORMANTS".

By means of certain hypotheses, we can consider the speech production system as a linear one-input one-output system, with an impulse response h(n). s(n) is the output at the lips and is given by the following convolution relation:

$$s(n) = u(n) \otimes h(n)$$

or in the Z plane

$$S(Z) = U(Z) \cdot H(Z)$$

and the amplitude spectra of the signals are as follows (f is the frequency):

Source amplitude spectrum: /U(f)/

Speech signal amplitude spectrum: /S(f)/

System transfer function : /H(f)/

The power spectrum of the speech signal is:

$$p(f) = /S(f)/^2$$

The CEPSTRUM of the speech signal is defined as the inverse fourier transform of " the power spectrum

expressed in dB". It is worth arranging the above relations summarized according to the time and frequency domains as shown in figure (4).

Figure (5) demonstrates two examples on these relations which are the cases of vowel and fricative sound production respectively.

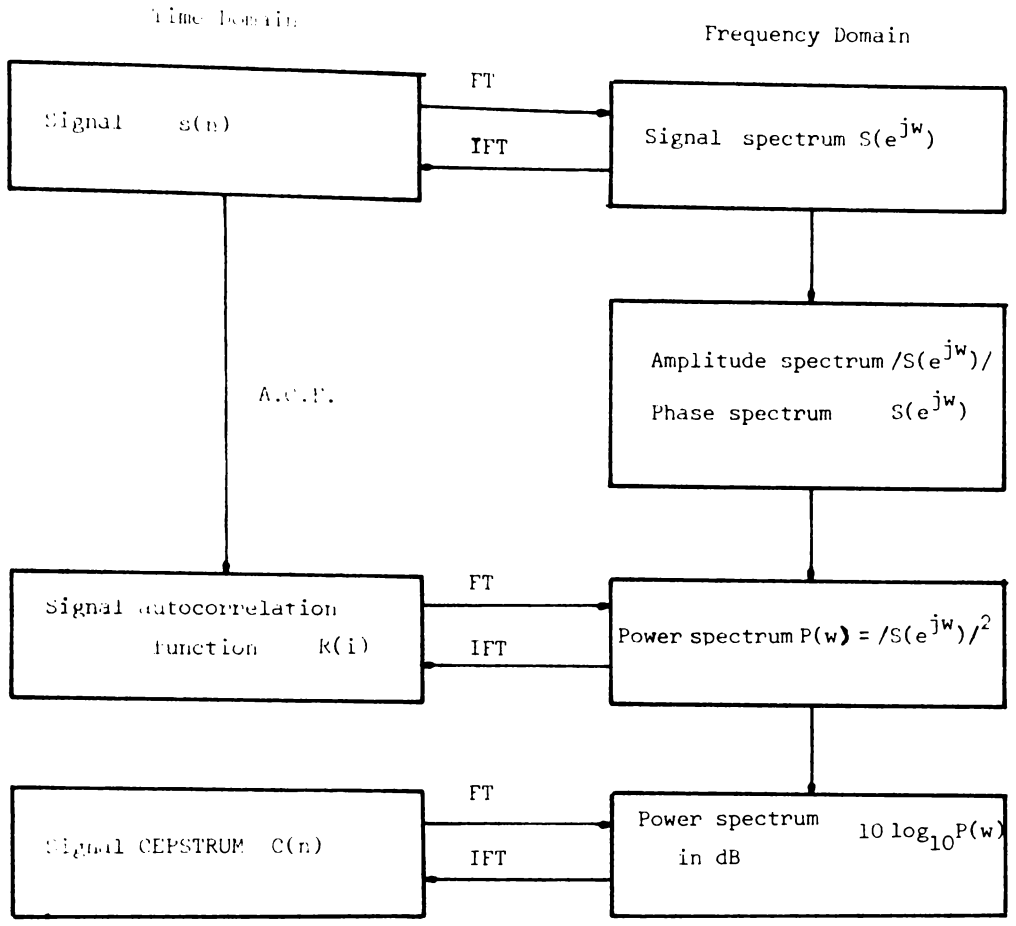


FIGURE (4) The principal processing operations on speech

FT : Fourier Transform
IFT: Inverse Fourier Transform
ACF: Autocorrelation Function

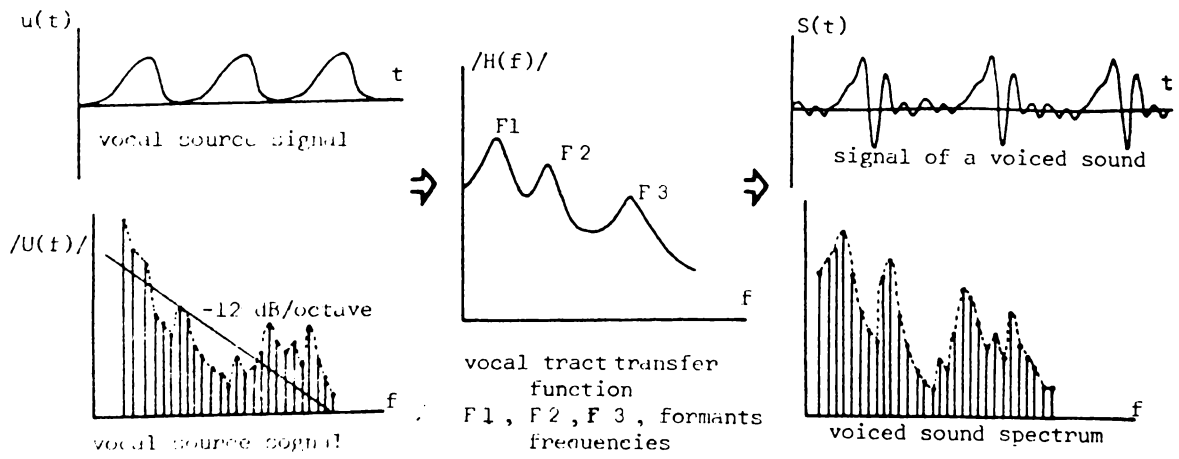


Figure (5.a): Production of voiced sound (vowels)

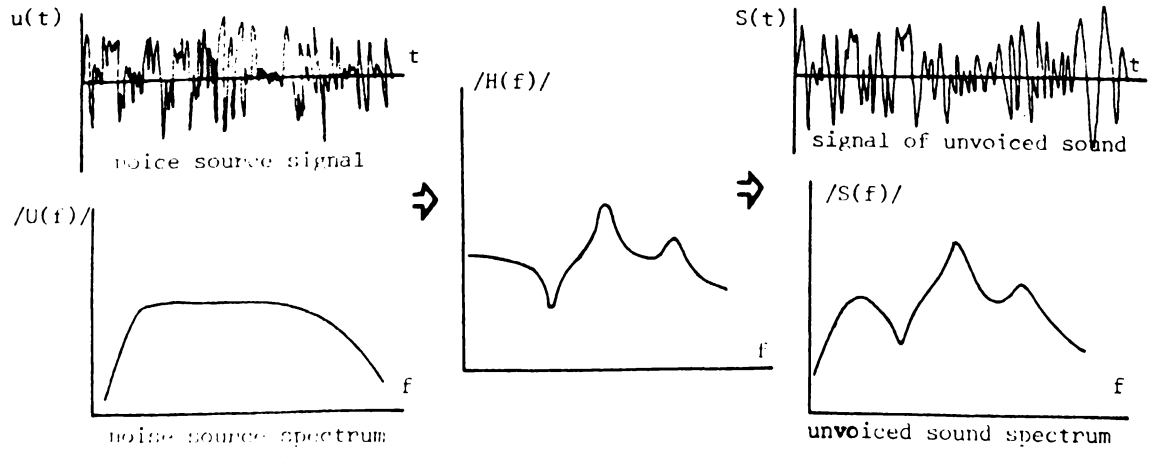


Figure (5.b): Production of unvoiced sound (unvoiced fricative)

In figure (6), we show an example of an Arabic phrase, "Scientific Studies and Research Center". In part (a) of the figure, the speech signal s(t) is shown, and in part (b) the three dimensional spectral representation of the signal is given: i.e the power in the signal as a function of frequency and its evolution with time; this representation is called a spectrogram. The fundamental frequency of the phrase is given in part (c) of the figure while the total energy of the signal is shown in part (d).

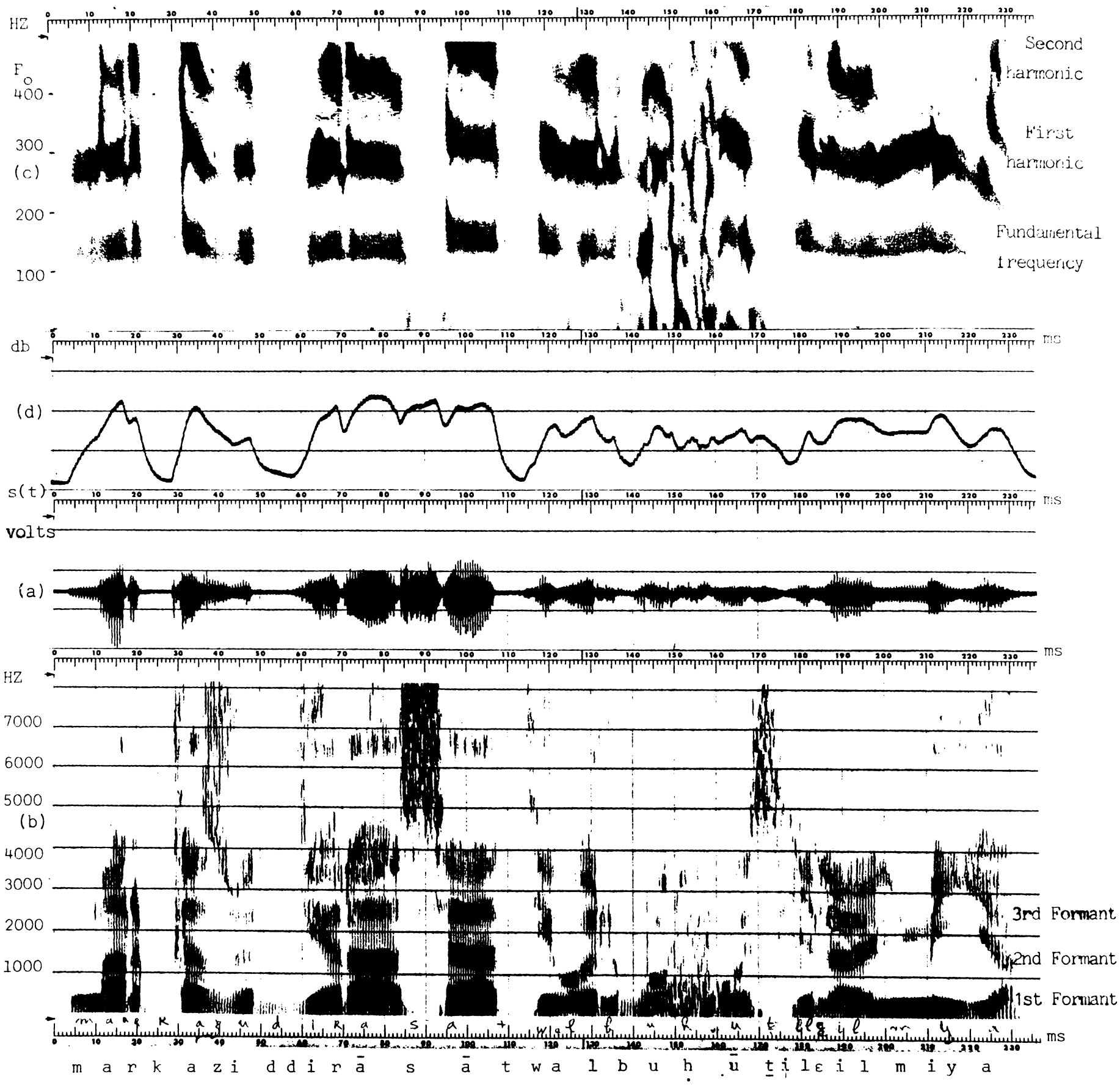


Figure (6): Analysis of the Arabic Phrase "Scientific Studies and Research Center"

a The temporal speech signal s(t).

b Spectrogram :Three dimentional analysis of the signal, power as a function of freq. and time

c Fine analysis of the lower frequecy band of the signal showing the fundamental frequency and its two harmonic for voiced sounds.

d Total energy in the signal as a function of time.

I.3 Speech (Arabic Phonetic System):

Speech is a coding of the acoustic signal which permits the transmission of information. This coding is realized by modifying the characteristics of the excitation sources and the supra-glottal tracts. The spoken message is redundant and includes specific information on the speaker which is not part of the coding system, for example : the melody of the voice, the spectral details of the speaker vocal source and nasal tract and the speaker emotional state ... etc.

The phoneme is defined as a language sound element characterized by a number of distinct pertinent features that distinguish it from the other phonemes of the language. The phoneme represents the elementary unit of the speech message on the phonological plan, but, on the acoustic plan, we study particular realizations of it. We can classify the standard Arabic language phonemes in two categories:

a) Vowels

b) Consonants

Standard Arabic has basically 35 phonemes which are six vowels and 29 consonants (the 29th is the pharyngealized /l̤/which is very rarely used).

a) The Vowels:

The vowels are produced when the vocal tract is excited by the periodic signal generated from the vibration of the vocal cords. They present a stable central part which can be considered as stationary. Standard Arabic has three short vowels /a/, /i/, /u/ and three long ones /ā/, /ī/, /ū/. We arranged these vowels in table (1) depending on the degree of constriction at the articulation point, and on the tongue hump position.

Table 1 : Arabic Vowel Classification

Tongue hump position / Degree of constriction	front	central	back
High	yā3-maddiyah ī kasra i		wāw-maddiyah ū ḍamma u
Low		Alif ā fatha a	

b) Standard Arabic Consonantic System [5] [6] :

Consonants could be divided into five main categories:

1) The fricatives which are produced when the vocal tract is excited by a noise source caused by a constriction at the place of articulation. Standard Arabic has 14 fricatives which are:
   - Voiced : / ɛ ',/ǧ/,/ǰ/,/Z/,/ḍ/,/Ẓ/;
   - Unvoiced : /h/ , /ḥ/, /x/, /Š/, /S/, /Ṣ/, /ṯ/, /f/ .

   and of which the consonants /Ẓ/ and /Ṣ/ are pharyngealized.
2) The stops which are produced when the vocal tract is excited by a noise source caused by an occlusion . We have 8 stops in standard Arabic which are:
   - Voiced: /d/,/ḍ/,/b/;
   - Unvoiced: /ɜ/,/k/,/ḳ/,/t/,/ṭ/;

   and of which the consonants /ḳ/, /ḍ/ and /ṭ/ are pharyngealized.
3) The nasals : which are produced when the vocal tract coupled to the nasal tract are excited by the vocal source (periodic signal). The vocal tract is closed at the place of articulation. Standard Arabic has two nasals /m/ and /n/.
4) The glides : /l/ and /r/ are produced when the vocal tract is excited by the vocal source. The place of articulation form a special constriction that permits wave propagation through both sides of the tongue.
5) The semi-vowels /y/ and /w/ (yā3 Al-līn and wāw Al-līn): They are characterized by a narrower constriction than that of vowels.

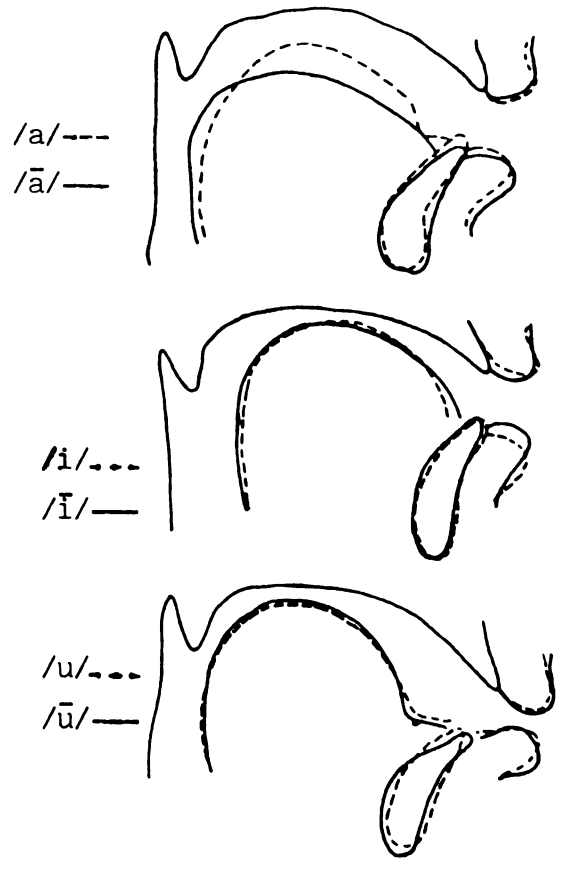


Figure (7): Articulation of Arabic short and long vowels AL-ANI-1970

Table 2: Tentative Chart of the Standard Arabic Consonantic System.

			Bilabial	Labiodental	Interdental	Alveolar Alveodental	Palatal	Velar	Uvular Pharyngeal	Pharyngeal	Glottal
Stop consonants	Voiced	Pharyngealized				ض d.					
			ب b			د d					
	Unvoiced	Pharyngealized				ط t.			ق q		
						ت t		ك k			ء '
Fricative consonants	Voiced	Pharyngealized			ظ ð.						
					ذ ð	ز z	ج ǧ		غ ġ	ع ʿ	
	Unvoiced	Pharyngealized				ص s.					
				ف f	ث θ	س s	ش š		خ x	ح ḥ.	ه h
Nasals	Voiced		م m			ن n					
Glides		Pharyngealized				ل l.					
						ل l ر r					
Semi-vowels			و w				ي y				

Table (2) shows a tentative chart of the standard Arabic consonantic system. This system differs from the Latin one primarily by the presence of the pharyngealized (emphatic) phonemes, the pharyngeal phonemes and the glottal phonemes.

II. SPEECH SYNTHESIS:

As we mentioned above, Electronic Speech Synthesis, ESS, is the production of spoken messages from a parametric transcription of speech.

For example: Text-to-speech synthesis is realized using the following steps:

1 - Orthographic to phonetic transcription of the message.

2 - Expressing the phonetic code of the message as a set of parameters and a set of rules. The nature of the parameters depends on the synthesis technique to be used while the rules depend on the synthesis method or strategy adopted and on the language under consideration. In addition to the technique of playing prerecorded speech, there are four main synthesis techniques used today, namely: channel vocoder, linear predictive coding (LPC), Formant and articulatory modeling. On the other hand, four synthesis methods can be adopted depending on the application; these are synthesis by sentences, words, diphones and rules.

3 - Using the above mentioned set of parameters and rules to generate appropriate control parameters suitable to the type of synthesizer used.

4 - Generating the speech message by sending the sequence of the control parameters, corresponding to the message, to the synthesizer.

We will review briefly the ESS techniques and methods.

II.1 E.S.S. Techniques:

Two major types of speech synthesis techniques can be distinguished which are:

a) Synthesis using pre-recorded speech elements. These techniques use digitized or wave form coded speech which range form the Hi-Fi 128 kbit/s to the PCM 64 kbit/s down to delta modulation speech at 16 kbit/s. [7] [8] .

b) Synthesis by re-constructing speech using parameters of a source-filter model which can be realized in the frequency or time domain. The bit rate involved in these techniques ranges from 50 bit/s for the articulatory model to 2400 bit/s for the LPC, up to channel vocoder at 9600 bit/s.

Table (3) shows a comparision between the different ESS techniques. It will be followed by a review of these four techniques.

Table 3: Principal Features of E.S.S. Techniques.

Features \ Synthesis Technique	Digital memory or tape recorder	Formant synth.	Channel vocoder synth.	Linear Predictive coding synth.	Synthesis by vocal tract simulator
Coding Parameters	Signal	3-variable formants+nasality+ amplitude+pitch	12-14 channels + excitation	Reflection coefficients+pitch	articulatory parameters
Quantity of information needed for synthesis	analog signal or 64 kb/s PCM 32 kb/s D-Mod	1000 b/s	2400 - 9600	1000-2400	Final goal is 50 b/s
Output-Speech intelligibility and naturalness	Excellent	Good	Good	Very Good	Expected to be Good
Hardware Complexity	Simple	Medium	Medium	Medium	Complex

II.1.1 The Channel Vocoder:

This technique was introduced by Dudley (1939). The principle behind this technique is, on the one hand, to find the power spectrum $/S(f)/^2$ of the speech signal as seen by an appropriate temporal window. This spectrum is divided into 12 to 24 bands. The evolution, with time, of the power in each band is considered as one parameter, which means that we can have from 12 to 24 such parameters depending on the quality desired. On the other hand, the fundamental frequency is detected when the type of source excitation is voiced. We also have to decide whether the sound is voiced or unvoiced. Consequently, if we have 14 bands C_1 to C_{14}, then the total number of parameters is 16. These parameters are used to synthesize speech using the configuration shown in figure (8).

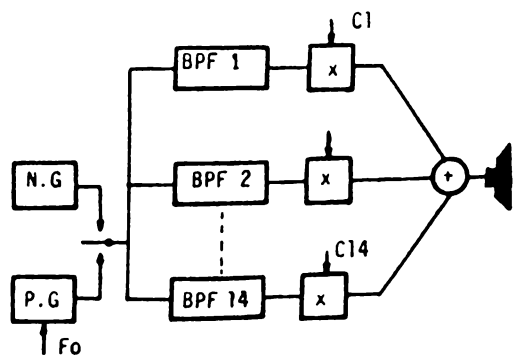


Figure (8): Channel Synthesizer

The typical bit rate needed to control a channel vocoder synthesizer ranges from 2400 to 9600 bit/sec.

The channel vocoder can be realized using either analog technique by means of a bank of analog butterworth band-pass filters, or digital techniques by means of digital filters or FFT algorithms.

The main advantage of this technique is that analysis is automatic. Many versions of this technique have been proposed. The drawback of the channel vocoder is the "mechanical or monotonic" aspect of the speech produced.

II.1.2 The Formant Synthesizer:

This technique was introduced by Fant in the early 1950's. It simulates the transfer function of the vocal tract H(Z). As mentioned already, the transfer function is characterized by several resonance frequencies, and it can be simplified by taking the first three formants F_1,F_2 and F_3 see figure [6] . The value of each formant changes from one phoneme to another. The transfer function of the nasal tract has fixed formants (N_1,N_2,N_3 and N_4) because, practically, the nasal cavity does not change its dimensions. But in addition to the formants the nasal tract presents one major antiformant (No). As shown in figure (9) the two above mentioned transfer functions are realized using two branches of filters, one to generate vowels and the other to generate nasals. These two branches are excited by a periodic source whose fundamental frequency, F_o, is controlled according to the melody of the synthesized message. In order to generate fricative sounds, a third branch is added with two noise formants B_1, B_2 and one noise antiformant B_o . This last branch is excited by a white noise source. The amplitude of the signal exciting each branch is controlled depending on the relative power in each branch. The total number of parameters needed to control this synthesis is ten. i.e.: F_o,A_n,A_o,A_{BF}, A_{B1},F_1,F_2,F_3,B_1,B_2 .

The typical bit rate needed to control a formant synthesizer ranges from 1000 to 2400 bit/ sec.

Formant synthesizers are realized using analog

filters [9] [10], or digital filters by simulation on a computer [11] [12].

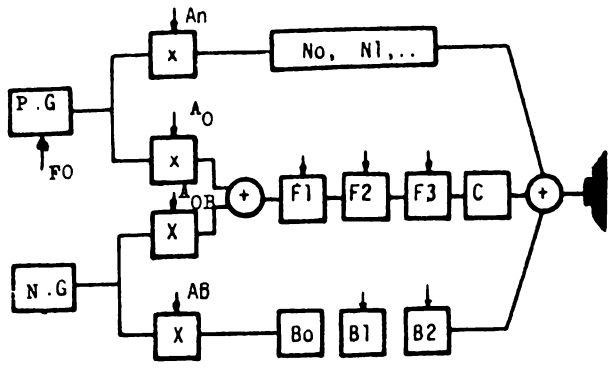


Figure (9): Formant Synthesizer

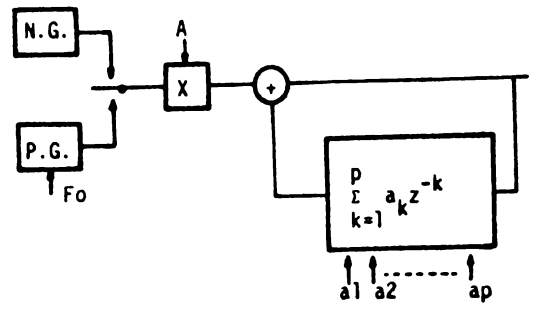


Figure (10): L.P. Synthesizer

The main advantage of this technique is its considerable degree of flexibility and efficiency in the various applications of synthetic speech. It provides important insight into the basic mechanism of speech production and perception. The drawback of this technique is the difficulty of reliable automatic analysis of speech to detect the formants.

II.1.3 The Linear Prediction Synthesizer:

This technique was introduced in the early 1970's by several researchers [15] [16] [17]. It became immediately very popular because its introduction coincided with the fast development of digital computers, and this technique lends itself easily to computer simulation and to implementation on integrated circuits.

Linear prediction is equivalent to all-pole modeling or autoregressive modeling of the speech signal. The speech sample S(n) is approximately predicted as a linear combination of a number of immediately preceding samples:

$$\tilde{S}(n) = - \sum_{k=1}^{p} a_k S(n-k)$$

Where a_k $1 \leqslant k \leqslant p$ are known as the predictor coefficients and p is the order of the predictor. It is possible to define an error e(n) between S(n), the true value of the signal, and $\tilde{S}(n)$ the predicted value i.e.,

$$e(n) = S(n) - \tilde{S}(n)$$

$$e(n) = S(n) + \sum_{k=1}^{p} a_k \quad S(n-k)$$

By taking the Z transform of this equation we get:

$$E(Z) = S(Z) \left\{ 1 + \sum_{k=1}^{p} a_k \; Z^{-k} \right\}$$

$$E(Z) = S(Z) \cdot A(Z)$$

It is clear that A(Z) is an all-zero filter and we can obtain e(n) by passing the signal S(n) through this filter. e(n) has a relatively flat spectral envelope.

The predictor coefficients are chosen so as to minimize the mean square prediction error $\overline{E(n)^2}$, averaged over all n; or in other words, to minimize the energy or variance of the residual e(n):

$$\overline{e(n)^2} = \sum_{n=1}^{\infty} \left[S(n) + \sum_{k=1}^{p} a_k \; S(n-k) \right]^2$$

We differentiate with respect to a_k and the result is set to zero giving the set of normal equations:

$$\sum_{k=1}^{p} a_k \; R(i-k) = - R(i) \qquad 1 \leqslant i \leqslant p$$

which is a set of p linear equations in p unknowns which is solved for the a_k. These a_k are the predictor coefficients. Since e(n) is white noise and E(Z) = A(Z) . S(Z), therefore:

$$S(Z) = \frac{E(Z)}{A(Z)} = \frac{1}{A(Z)} \cdot E(Z) = H(Z) \cdot E(Z)$$

and the signal spectrum can be modeled by the spectrum of the all-pole filter H(Z). For synthesis the system shown in figure (10) is used to give good quality. The parameters needed for synthesis are the predictor coefficients, the fundamental frequency, the voiced/unvoiced decision and the gain.

* The typical bit rate needed to control an LPC synthesizer ranges from 1000 to 4800 bits/s.
* The LPC synthesizer is very easily realizable using computer simulation or digital circuits.
* The main advantage of this technique is its suitability to digital realization and the good speech quality obtained using 1000 bits, and finally the possible automatic analysis of speech to obtain the parameters. The drawback of this technique, as in the channel vocoder, is that the choice between either periodic or noise sources is not always satisfactory and recently several attempts have been made to solve this problem [16] [17].

II.1.4 Articulatory Synthesis:

This technique was introduced in its simplest form by Dunn 1950 [18] and Stevens et.al. 1953 [19] . The vocal tract is represented by a lossy acoustic tube of variable cross section A(x,t), having non-rigid walls. The propagation of sound waves inside this tube is simulated by solving the partial differential equations describing the physical phenomena covering this propagation. One form of these equations is [3] :

$$-\frac{\partial p}{\partial x} = \rho_0 \frac{\partial}{\partial t}\left(\frac{U}{A_0}\right)$$

$$-\frac{\partial U}{\partial x} = \frac{\partial A}{\partial t} + \frac{1}{\rho_0 c^2}\frac{\partial (pA_0)}{\partial t}$$

$$p - b_p \frac{dr}{dt} = M_p \frac{d^2 r}{dt^2}$$

$$A(x,t) = A_0(x,t) + S_0(x,t)\ r(x,t)$$

These equations can be solved using different approaches. One simple approach is to consider the tube as a succession of short uniform cross section tubes . The cross sectional area of each tube varies with time {see figure 11} .

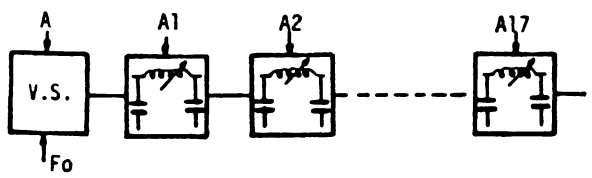


Figure (11): Vocal Tract Simulator

Another approach is to transform these equations into finite partial difference equations, and to solve them numerically. The variables x and t are sampled in the plane (x,t) (i.e. spacially and temporally).

The actual parameters used in this technique are the values of the air function A(x,t) corresponding to the message to be synthesized. This function could be deduced from an articulatory model using parameters such as : tongue position , jaw position etc. These parameters change relatively slowly with time and could be coded by a bit rate as low as 50 bits/sec.

Articulatory synthesizers have been realized employing analog or digital techniques [20] [21] [3] .

The quality of synthetic speech obtained can be almost equivalent to natural speech for certain types of sound such as vowels. This technique offers a valuable tool for fundamental research on speech [22] [23] [24] [25] [26] .

The difficulties of this technique are the complex rules required to generate the control parameters and the non-linearity involved in simulating the noise source for synthesizing the frictives. Extensive research is required on the articulation of particular standard Arabic consonants such as the glotal, pharyngeal and pharyngealized ones.

II.2 ESS Methods [27][28]

There are four synthesis methods:

a - Pre-recorded or pre-analysed sentences.

b - Pre-analysed words.

c - Diphone.

d - By rules.

Each method can be used with several techniques, but usually is more adapted to one of the techniques. For example, synthesis by rules is often used with Formant synthesis technique. In each of the four methods, we have two major blocks which are the "dictionary" and the "set of rules". In the dictionary, one stores the "building blocks" which are concatenated together using the "set of rules" to generate the set of parameters corresponding to the synthesized message. The size of the " dictionary " is inversely proportional to the size of the "set of rules". The four ESS methods will be reviewed below.

II.2.1 Synthesis by Pre-recorded or Pre-analysed sentences:

For a given application where the needed synthetic messages are limited, one can prerecord and store digitally the desired sentences; the "synthesis" operation in this case is digital to analog conversion. This method requires a large memory capacity and is not economical. A better method is to analyse these sentences and store them as a sequence of sets of parameters in accordance with the synthesis technique to be used. This method requires a smaller memory capacity (30 - 60 times less). In these two methods, one does not need any set of rules or any rules of prosody and the synthetic speech quality is very good.

II.2.2 Synthesis by Pre-analysed Words [29]

This method is based on the storage of speech elements representing full words or parts of sentences . The elements are stored as the sequence of parameters obtained by the analysis of these elements using one of the four techniques mentioned before. The size of the dictionary depends on the application and is limited to it. To synthesize a message, access is made to the appropriate sequences of parameters which are concatenated together. By means of prosody rules,

the intonation of the message is generated. The quality of synthesis can be obtained and the number of rules needed is very small.

II.2.3 Synthesis by Preanalysed Diphones [30][31]

In order to be able to synthesize unlimited vocabulary, the dictionary should be composed of basic phonetic elements from which words and sentences can be composed. Simple concatenation of phonemes yields speech which sounds not only unnatural, but is unintelligible [32] . This is because of the fact that phonemes are affected by each other and because the transition between phonemes is very important for the intelligibility [33] . One simple solution is to use the diphone as the basic phonetic element. The diphone is the segment which lies between the steady state portion of one phonetic realization, and that of the next phonetic realization. This definition is given in the acoustic domain, to avoid ambiguity with the abstract phonological notion of phonemes. However, this definition is not very precise, since the steady state portion of a phonetic realization does not always exist.

The dictionary, in this methods, is composed of the diphones of a language which are usually of the order of 1000 and the duration of each is approximately 100 to 200 ms. By means of analysis, using LPC, Formant or vocoder techniques, the sequences of sets of parameters corresponding to these diphones are obtained and stored. Simple rules for concatenating the diphones are also stored. Consequently this method is characterized by a medium size dictionary (30 - 1000 kbytes) and a small number of simple rules. The quality of speech obtained is acceptable for many applications. Prosody rules are also necessary in order to achieve this quality.

II.2.4 Synthesis by Rules:

The second method used to synthesize unlimited vocabulary is synthesis by rules. The basic speech elements in this method are phonemes, allophones or subphonemic units. The size of the dictionary is small (1 to 8 kbytes). Figure (12) shows an example of this method given by Liberman et al. [33] .

The number of rules necessary to generate the sequence of sets of parameters, representing the succession of sounds to be synthesized, is very high and complex. These rules can operate on different types of sets of parameters depending on the technique used. It has been realized for the Formant parameters [34] [35][36][37] , and the articulatory parameters 38,but, naturally, it is not feasible by channel vocoder parameters. Prosody rules are also required. This method enables the simulation of male or female voices.

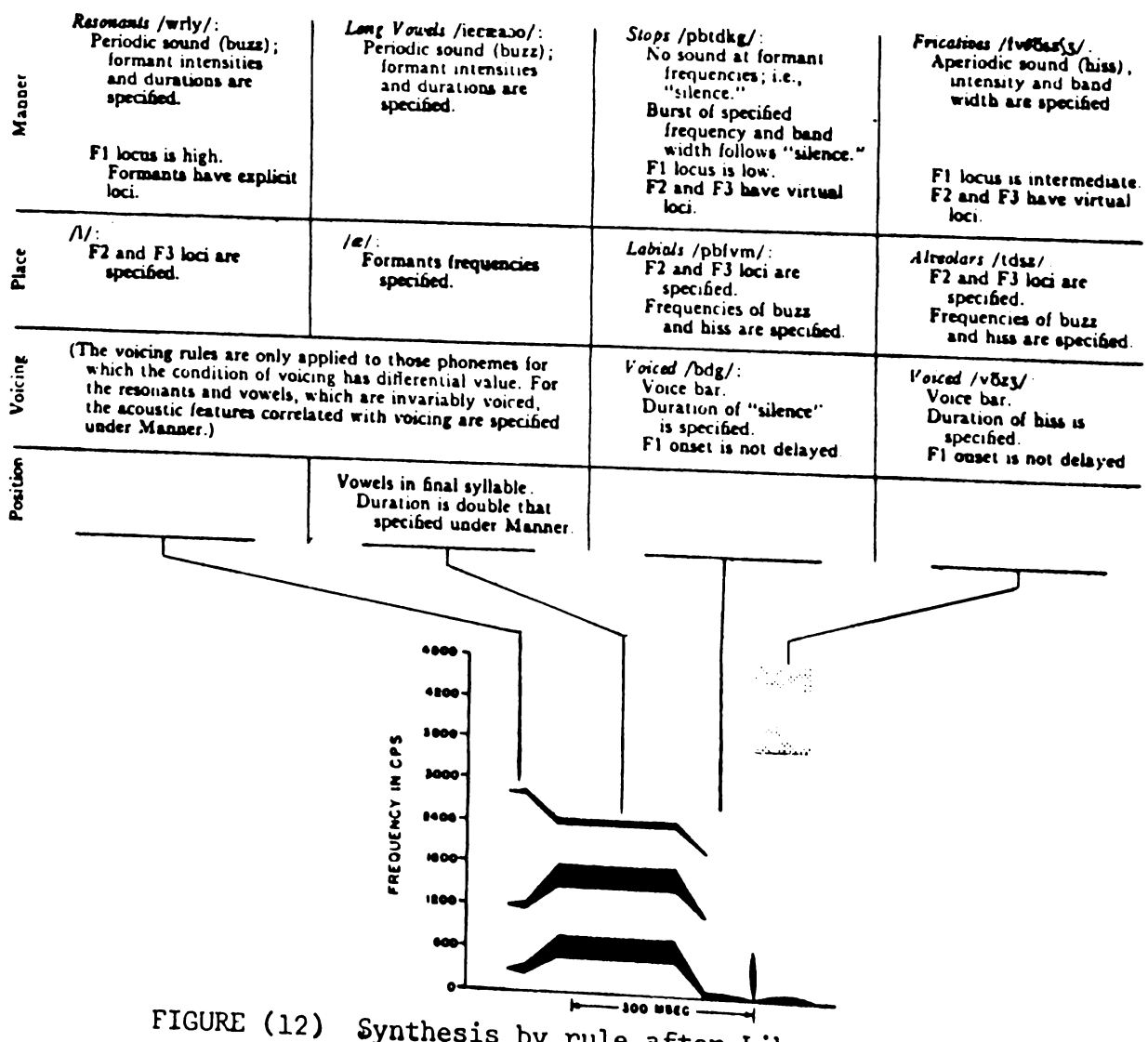


FIGURE (12) Synthesis by rule after Liberman & al, 1959

Finally, table (4) shows a comparison between the four above mentioned methods.

Table 4: Comparison between ESS methodes.

Features / ESS Method	Synthesis vocabulary	Speech quality	Voice type (male or female)	Size of dictionary	Number of rules	Size of prosody rules
Pre recorded or Pre analysed sentences	limited to given application	Very good	One type	(large) depends on the vocabulary	None	None
Pre analysed words	limited to given application	Good	One type	(large) depends on the vocabulary	Very small	Small
Pre analysed diphones	unlimited	Sufficient for many applications	One type	(medium) 30-100 Kbyte	Small and simple	Medium
Rules	unlimited	Good	Both types	(small) 1 - 3 kbyte	Very high and complex	Medium

II.3 Knowledge Required on Arabic Language for ESS:

Table (5) shows the four speech synthesis methods and the five synthesis techniques with indication of their language dependency (L.D.). As we can see, it is possible and interesting to generate synthesized Arabic speech using digital storage of sentences, which is non-language dependent (N.L.D.). However , this would be very expensive in systems having a large number of sentences; and impossible for unlimited vocabulary systems. All other synthesis methods are language dependent, no matter what technique is used, and this dependency is very important when we deal with synthesis by diphone and even more important when we employ synthesis by rules. Applied research in this field on the Arabic language is necessary and it should start by the L.D. areas. There are many applications, where information from a computer or a "machine" is to be presented in its spoken form. The possibility of using the Arabic language is a prerequisite for the social and economical acceptance of these applications. (e.g. computer spoken output, sophisticated alarm systems, automated announcements systems, vocal-interactive training systems ... etc.).

Examining the methods and techniques mentioned above, we can deduce that for the installation of such systems employing the Arabic language, research is required in the following specific domains:

a) Detailed description of the phonetic system of standard Arabic .

b) Rules for orthographic-phonetic transcription of standard Arabic.

c) Data and rules on the prosodic features of standard Arabic and the rules for their generation :

- micro and macro-melody
- duration
- stress

d) Dictionaries and rules for the generation of parameter values corresponding to the adopted synthesis method and technique, e.g.,

- Formant values and the rules governing their transitions .
- Diphone formant values
- Diphone LPC (reflection coefficient) values
- Spectral analysis of diphones for channel vocoder technique
- Articulatory parameters or A(x,t) values and rules governing their dynamics

Finally, several efforts are being made in the Arab World to achieve ESS using diphones and LPC techniques [39] [40] . The following examples of Arabic speech synthesis will be demonstrated:

- Synthesis by preanalysed sentences using Formant technique
- Synthesis by preanalysed sentences using LPC technique
- Synthesis by preanalysed sentences using channel vocoder technique

- Synthesis by diphone using Formant technique.
- Synthesis by diphone using LPC technique.

Table 5 : Speech Synthesis Methods and their Language Dependency

ESS Techniques / Synthesis Methodes	Digital memory or tape recorder	Format synth.	Channel vocoder synth.	LPC synth.	Synthesis by vocal tract simulator
Sentence	N.L.D	N.L.D	N.L.D	N.L.D	N.L.D
Word	L.D	L.D	L.D	L.D	L.D
Diphones	not possible	V.L.D	V.L.D	V.L.D	V.L.D
Rules	not possible	E.L.D	not possible	E.L.D	E.L.D

L.D. : Language dependent, N.L.D. : Non-language dependent, V.L.D. : Very language dependent.
E.L.D. : Extremely Language dependent .

III. CASE STUDY: TEXT TO SPEECH SYNTHESIS OF STANDARD ARABIC:

A block diagram of a text to speech synthesis system is shown in figure (13) [31] . This system could also be used as a computer vocal response system and in most machine to man communications systems.

The system is being realized using a VAX-11/780 computer as a research tool but could be installed on micro-computers.

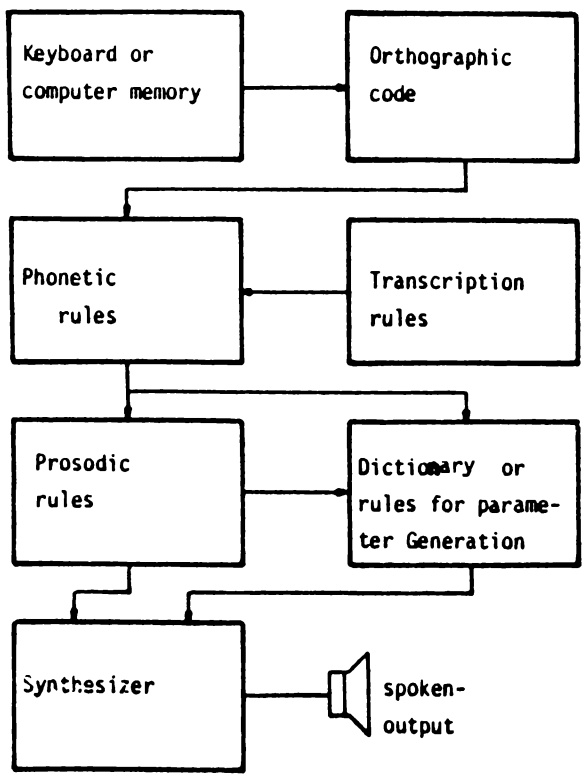


Figure (13): Block diagram of a computer vocal response system

An LPC synthesis technique is adopted with the diphone method. The dictionary of diphones will contain more than 1500 diphones.

A program to realize the orthographic-phonetic transcription has been written for standard Arabic. The input to this program is the 28 Arabic characters plus the diacritics and a number of punctuations and prosodic symbols, while the output is the sequence of diphones corresponding to the text being synthesized plus the prosodic information.

The prosodic rules include knowledge about the macro melody, stress and duration. Standard Arabic has nineteen types of sentences. We have analysed the four most frequently used and some general rules have been deduced.

Real time synthesis is achieved by means of a hardware LPC synthesizer interfaced to the VAX. The terminal used with this system is Questar.

IV. ORTHOGRAPHIC TO PHONETIC TRANSCRIPTION:

The transformation of grapheme strings to phoneme strings is language dependant.

In general, Arabic text with diacritic is pronounced as it is written using certain rules. There are few exception to this principle. Contrary to English, Arabic does not have words with different orthographic forms and the same pronunciation.

There are eight essential rules in orthographic to phonetic transcription. These rules are used in a computer algorithm with other rules as an input to the prosodic feature generating program. Each of the eight rules is presented briefly in the following

paragraph.

1 - 3al-lām 3al-kamariya (lunar-lām) and 3al-lām 3aš-šamsiya (solar-lām):

The letter lām " ل " in (الـ) is pronounced if followed by one of the fourteen following characters: (ا ، ب ،غ، ح، ج، ك، و، خ، ف ،ع، ق، ي، م، ه), and is assimilated with the succeeding character if it is one of the fourteen following characters : (ل، ش، ز، ظ، س، د، ن، ذ، ض، ر، ت، ص، ث، ط).

Examples:

The " lām " is pronounced in: القمر ، الحجر ، الولد

and not pronounced in: الشمس ، الصخر ، النور

2 - Pharyngealization (emphasis):

The pharyngealized consonants in standard Arabic are the following (ظ ط ض ص) . In addition, the ر ل and ا are also pharyngealized in certain cases such as:

a) The ر is pharyngealized in three situations:
   - When followed by the short vowels /a/ or /u/ .
     Example: رَمَقْ - رُقِي
   - When preceeded by the short vowels /a / or /u/ and not followed by a short vowel (sākinah).
     Example: غُرفَة - زُرافَة
   - When it is sākinah and preceeded by /i/ and followed by one of the 3istiɛlā3 consonants which itself is not followed by /i / .

b) The 3alif " ا " is pharyngealized when it follows one of the 3istiɛlā3 consonants or the pharyngealized ر and ل .

c) The lām " ل" is pharyngealized in the word " الله " when it is preceeded by /a/ or/ u/.

3 - hamzat-3al-wasl

This " hamza " is pronounced when it comes after a pause or at the begining of an utterance. It is not pronounced in all other cases. This hamza is found in the following patterns:

a) In the imperative of triradical verbs :
   Example: اِعْلَمْ

b) In the past tense, imperative and masdar of five-consonant and six-consonant verbs:
   Example:
   Five-consonant: اِنْطَلَقَ - اِنْطَلِقْ - اِنْطِلاق
   Six-consonants: اِسْتَغْفَرَ - اِسْتَغْفِرْ - اِسْتِغفار

c) In the following nouns:
   ابن - ابنه - امرؤ - اسم - ا ست - اثنين - اثنتين - ايمن - امرأة .

d) In " الـ " 3attafrīf

4 - The long vowels: " حروف المد "

These are the three long vowels which are 3alif , wāw , and yā3 (ا ، و، ي). We give below four main rules governing special cases of their pronounciation:

a) The long vowel is not pronounced (changed to a short one) when it comes at the end of a word which is followed by another word starting by a non-vowelized consonant.

b) The long vowel is not pronounced (changed to short one) when followed by tanwīn.

Example on (a) and (b):

Orthographic	Phonetic
اذا القلم	إِذَلْقَلَمْ
في البيت	فِلْبَيْتِ
أبو العلاء	أَبُلْعَلَاء
فتىً	فَتَنْ

c) There are a number of special case words in Arabic when the long vowels are written but not pronounced and these are:
   عمرو - أولئك - أولو - صائه
   Added to these words is the 3alif-3attafrīk which follows wāw-3al jamāɛah which is also written but not pronounced :
   Example: جلسوا

d) There are a number of special case words where long vowels are pronounced but not written and these are:
   (and its derivatives) الله - اللهم - لكن - هذا
   الرحمن - السموات - اله - أولئك .

5 - hā3 3atta3nīt

This hā3 is used in Arabic at the end of a noun to modify its gender from masculine to feminine .
Example: خديجة - نشيطة - علامه
This hā3 is pronounced as a hā3 if followed by a pause, otherwise it is pronounced as a tā3 .

6 - 3iltikā3 3assākinayn

This concerns the rules of pronouncing two successive words, when the last character of the first word and the first of the second word are not vowelized. The general rule is that the short vowel / i / should be introduced after the last character of the first word.
Example: خُذْ الْكتاب is pronounced خُذِ لْكتاب
عَميقُ البحر is pronounced عميقُنِ لْبحر

There are two exceptions to this rule :

a) When the first word is " مـن" and the second starts with " الـ", the short vowel / a /and not/ i /is introduced.
Example: مِنْ المكتـب is pronounced مِنَ لْمكتب

b) When the first word is terminated by mīm 3al ǰam ع and the second starts with " الـ" the short vowel /u/ and not /i/ is introduced.
Example:
لـهـمْ الغُرفـة is pronounced لَهُمُ لْغرفـة
مِنْهُمْ الموظفون is pronounced مِنْهُمُ لْموظفون

7 - The pause:

An utterance in Arabic is never terminated by a short vowel (ḥaraka).
Example:

Orthographic	Phonetic
الطالب نشـيطٌ	الطالب نشـيطْ
كان الطالب نشيطاً	كان الطالب نشيطاْ
غريبـــة"	غريبـــهْ
طالبـــة"	طالبـــهْ

8 - 3at-tanwīn

The rule for transcribing the three diacretic signs ـً , ـٌ and ـٍ are as demonstrated by the following examples:

Orthographic	Phonetic
قلـمٌ أحمـر	قَلَمُنْ أحمـر
كتاباً أخضـر	كِتابَنْ أخضر
معلمٍ قديـر	مُعَلِمِنْ قديـر

Finally, using these essential eight rules with some other minor ones, we have found an algorithm which performs the transcription grapheme-to-phoneme in real time for text-to-speech synthesis.

V. RESULTS ON ARABIC INTONATION [41] [42] [43]

The three main prosodic features needed for Arabic speech synthesis from text to speech are intonation, duration and intensity. In this paragraph, results on **Arabic** intonation is given. The melodic curve of a sentence can be considered as the combination of two curves, which represent the macro-melody, at the sentence level, and the micro-melody, at the word and syllabic levels. The macro-melodic curve depends on the type of the sentence being synthesized. A survey of Arabic sentence types is given with results on the macro-melody of some of them.

According to the majority of Arab grammarians, there are two types of sentential structures which are: Nominal structure and Verbal structure, syntactically, the Nominal structure starts with what the grammarians called Musnad 3ilayhi which could be a sentence or a noun phrase. The Musnad 3ilayhi is followed by a sentence (which might be Verbal structure, nominal equational structure, or NP-predicate) which is called Musnad. On the other hand, the Verbal structure consists of the Musnad followed by Musnad 3ilayhi. Both the Nominal and the Verbal structures could have a syntactic adjunct which is called Fadlah. These basic structures can accept the addition of definite syntactic categories giving a number of semantically different structures. The Arab grammarians consider two types of sentences from the semantic point of view. These are the xabariyah and the 3insa3iyah. The xabariyah sentence has three types, which are 3itbāt, nafī and ta3kīd. The 3insa3iyah sentence has also three types talab, šart and 3ifsāh. Table (11) shows the semantical classification of the Arabic sentence.

We have studied the intonation of the types the most frequently used. We give below results on four of these types which are encircled on table (11), namely: affirmative, exclamatory, interrogative with 3adāt and negative.

Five speakers of standard Arabic were chosen, four women (F1 , F2, F3 and F4) and one man (M1) . Each of them pronounced a continious text which was recorded and analyzed to find the melody using the pitch analyzer : VISIPITCH. The average value (X1) of the melody of each speaker is calculated and found to be as follows:

X1 equals for F1 236 Hz
F2 253 Hz
F3 245 Hz
F4 217 Hz
M1 108 Hz

We choose 12 sentences for each of the four types, that is, a total of 48 sentences. These 48 sentences were recorded for each of the five speakers totaling 240 sentences. These 240 sentences were analyzed using the VISIPITCH, and results, as shown on figure (14), were obtained. These results give the variation of the fundamental frequency of each sentence. To study the macro-melody of each sentence type, the attack, maximum and final values of the fundamental frequency for each sentence were found . These values were normalized using X1 for each speaker and averaged over the 12 sentences for each sentence type. **Tables (6) (7) (8) (9)** show the results ob-

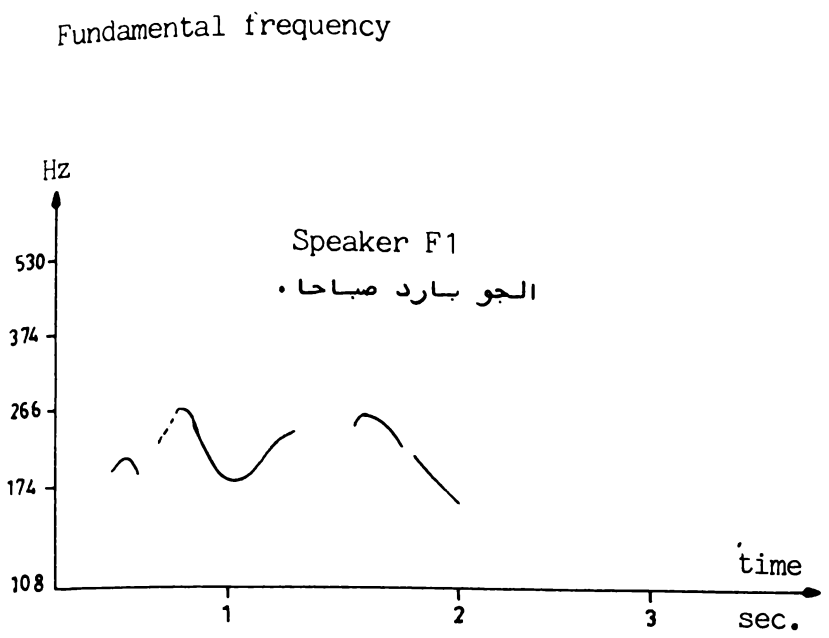


Figure (14-a) Melody of a Nominal affirmative sentence.

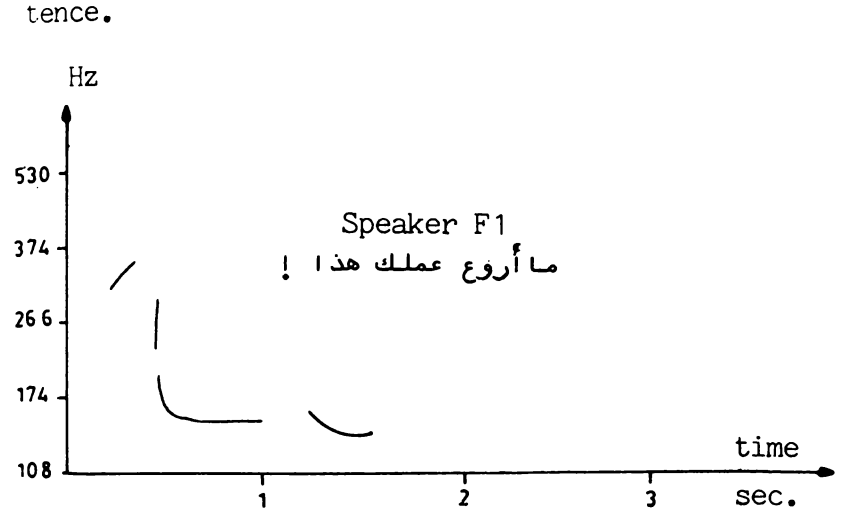


Figure (14-b) Melody of an exclamatory sentence.

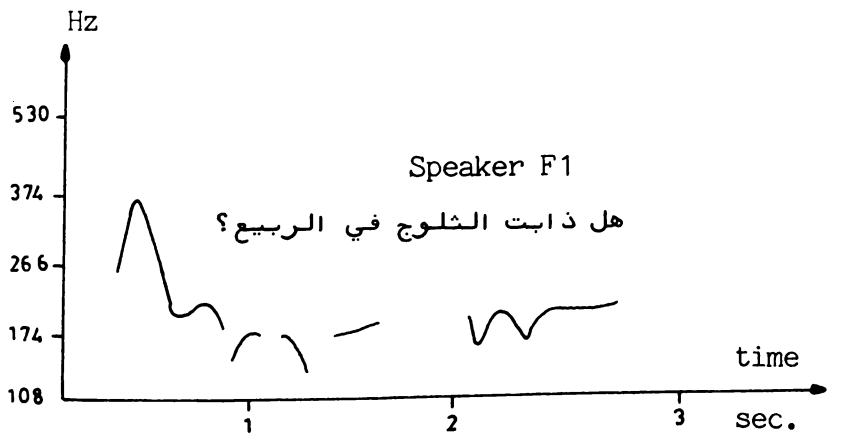


Figure (14-c) Melody of an Interrogative sentence with 3adāt.

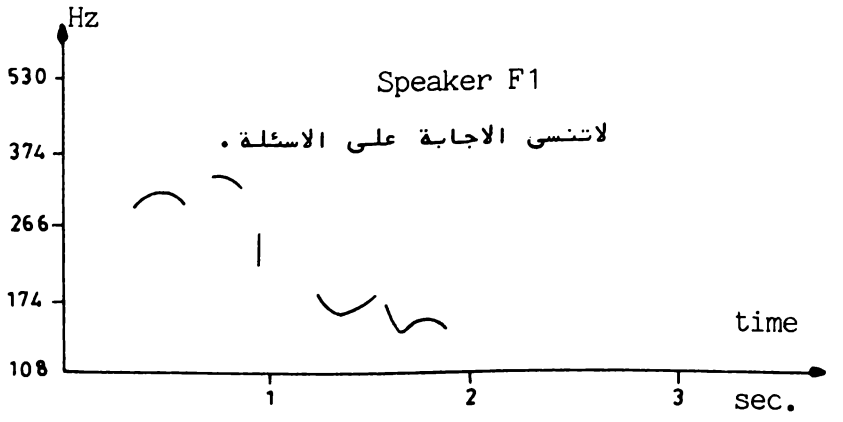


Figure (14-d) Melody of a Negative sentence.

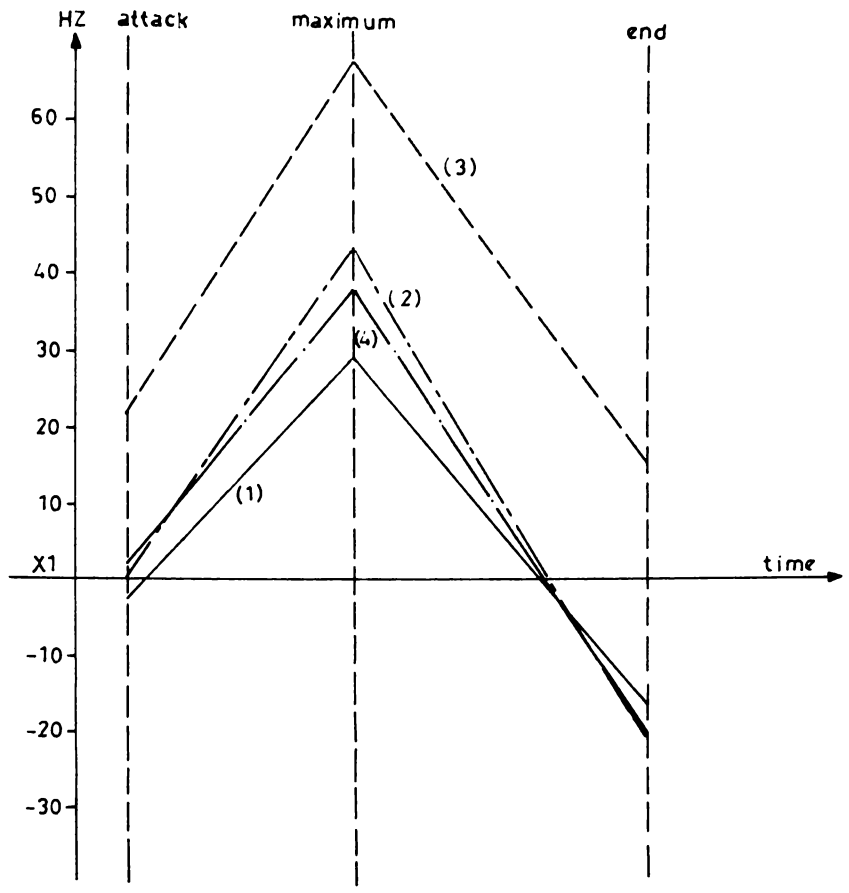


Figure (15) Normalized macro-melodic curves for the four sentence-types, averaged over five speakers and 12 sentences.

X1 is the average fundamental frequency.

(1) Affirmative;

(2) Exclamatory;

(3) Interrogative with 3adāt;

(4) Negative.

tained. Table (10) shows the final result which gives the average and standard deviation for the attack, maximum and final values for each sentence type averaged over all speakers. From this table and the other results, one deduces the following remarks (see figure (15)) .

1 - As it is already found for other languages, the melodic curve for the 4 sentence types show the presence of a declination line, a base, and a ceiling (plateau) which define a field of variation (Gamme).

2 - The interrogative sentence has the highest curve and highest maximum.

3 - The affirmative sentence has the smallest field of variation arround the average while the exclamatory one has the largest field which confirms the already established rule for other languages namely:

$$\text{Field of variation} \propto \frac{\text{emotion x anger}}{\text{sadness}}$$

4 - The interrogative sentence has a high attack and high termination.

Table (6) The normalized values of the frequency of attack, maximum and final points on the macro-melodic curve. Values are average over 12 sentences for each Speaker. Nominal affirmative sentence. σ is the standard deviation.

parameter / Speaker	att-x1 / x1	max-x1 / x1	fin-x1 / x1	σatt. / x1	σmax. / x1	σfin / x1
F1	-8,88	7,15	-19,1	8,75	8,09	8,51
F2	-6,75	38,34	-15,09	7,88	16,92	5,73
F3	-5,51	40,95	-22,55	9,18	11,81	10,5
F4	-1,54	19,82	-17,86	7,44	3,74	5,5
M1	8,02	38,58	-8,8	12,61	17,37	32,05

Table (7) The normalized values of the frequency of attack, maximum and final points on the macro-melodic curve. Values are average over 12 sentences for each Speaker. Exclamatory sentence.

parameter / Speaker	att-x1 / x1	max-x1 / x1	fin-x1 / x1	σatt. / x1	σmax. / x1	σfin / x1
F1	20,93	41,16	-18,08	13,2	19,3	3,15
F2	-2,93	59,35	-18,35	17,22	13,38	12,03
F3	-17,07	30,34	-29,29	10,18	11,22	18,17
F4	4,57	34,52	-22,66	11,55	6,13	9,74
M1	-1,39	50,46	-16,9	8,72	5,91	15,05

Table (8) The normalized values of the frequency of attack, maximum and final points on the macro-melodic curve. Values are average over 12 sentences for each Speaker. Interrogative sentence with 3adāt.

parameter / Speaker	att-x1 / x1	max-x1 / x1	fin-x1 / x1	σatt. / x1	σmax / x1	σfin / x1
F1	35,87	59,55	14,59	21,76	22,67	21,5
F2	19,76	80,6	19,07	12,5	21,39	22,02
F3	22,93	61,29	2,24	32,48	23,61	16,15
F4	9,79	36,87	2,15	9,93	17,94	12,52
M1	22,38	65,59	42,36	20,04	17,59	21,27

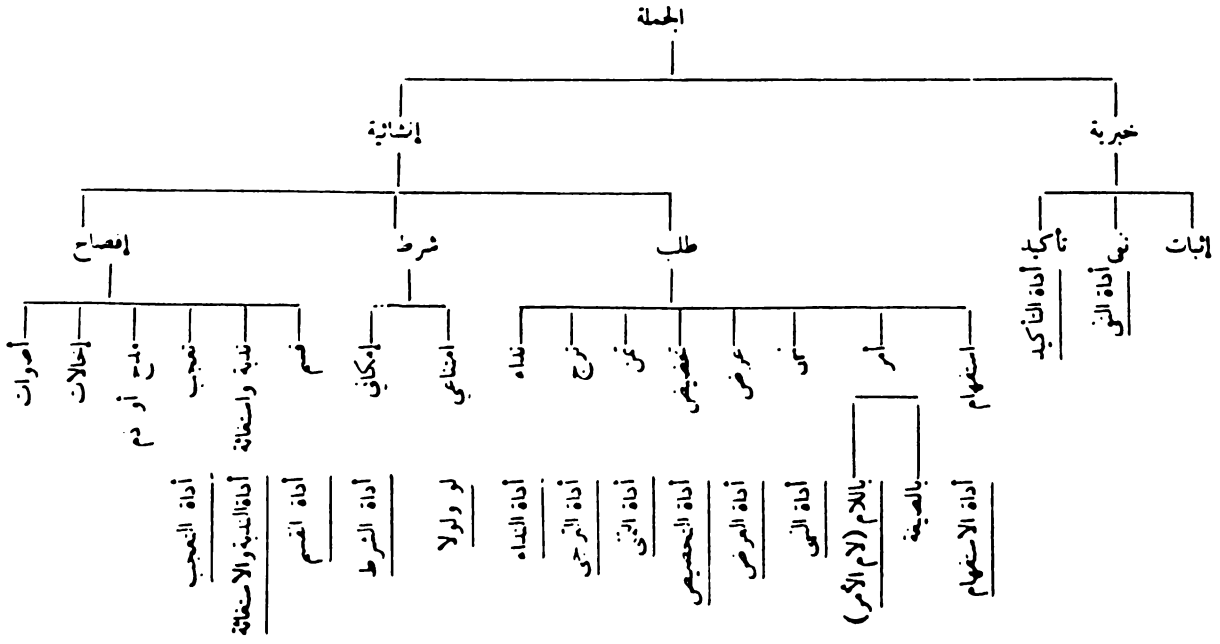
Table (9) The normalized values of the frequency of attack, maximum and final points on the macro-melodic curve. Values are average over 12 sentences for each Speaker. Negative sentence.

parameter / Speaker	att-x1 / x1	max-x1 / x1	fin-x1 / x1	σatt. / x1	σmax. / x1	σfin / x1
F1	18,57	41,51	-21,14	13,51	44,55	4,27
F2	-0,4	39,13	-14,86	5,92	10,48	10,58
F3	-7,14	38,5	-21,53	3,09	8,89	19,17
F4	2,61	21,12	-18,59	4,82	11,35	8
M1	-1,85	50,77	-26,08	4,14	5,83	9,78

Table (10) For each sentence type, the normalized values of the frequency of attack, maximum and final points averaged over the five Speakers.

parameter / Sentence type	$\overline{att-x1}$ / x1	$\overline{max-x1}$ / x1	$\overline{fin-x1}$ / x1	$\overline{\sigma att}$ / x1	$\overline{\sigma max}$ / x1	$\overline{\sigma fin}$ / x1
Nominal affirmative	-2,93	28,97	-16,68	9,17	11,59	12,46
Exclamatory	0,82	43,17	-21,06	12,17	11,19	11,63
Interrogative	22,15	60,78	16,08	19,34	20,66	18,69
Negative	1,64	38,21	-20,44	6,30	16,22	10,36

Table (11) Semantical types of standard Arabic sentences.



5 - The maxima are arranged as follows: interrogative, exclamatory, negative and then affirmative.

6 - As shown in figure (15),a first approximation of the macro-melody could be **a** linear curve connecting the three points. An exponential curve could also be used for synthesis purposes.

All terminations are lower than the average except for the interrogation sentence.

Finally, an extensive analysis of the different sentence types was performed using the sonagraph. documents as shown in figure (6) were found for each sentence. From these results the places of the maxima were determined for each sentence type. For example:

- The maximum in the affirmative Nominal sentence is on the second syllable of the Mubtadaɜ (subject).
- The maximum in the exclamatory sentence is on the first syllable of the verb following the ɜadāt of exclamation.

These results, coupled with other results on duration rules and other rules are to be used in a system to generate prosodic features from **a** text.

ACKNOWLEDGEMENT

Throughout the work on Arabic orthographic to phonetic transcription and on Arabic syntax, the author has enjoyed a close collaboration with Mr. M. Bawab, Mr. Y. Mer-Alam and Mr. H. Tayan. The work on the melody of sentences of standard Arabic is achieved in collaboration with Mrs. Hayder.

Finally, a personal note of gratitude goes to Mr. M. Farah for reading the proofs of the paper.

REFERENCES

[1] - Note Technique, CENT, NT/LAA/79 - August 1981 .

[2] - KLEIMAN, H. "Synthetic Speech: Explosive Growth ahead?" Speech technology, April 1982 .

[3] - MRAYATI, M. "Contribution aux études sur la production de la parole. Modèles électriques du conduit vocal avec pertes, du conduit nasal et de la source vocale. Etude de leurs interactions. Relation entre disposition articulatoire et caracteristique acoustiques". Thèse Docteur D'Etat INPG. Grenoble 1976.

[4] - FANT, G. Acoustic Theory of Speech Production , Mouton and co., S'Gravenhage. 1960 .

[5] - AL-ANI. S.H. Arabic Phonology. An Acoustical and Physiological Investigation, Mouton, 1970 .

[6] - GHAZALI, S. "Elements of Arabic Phonetics" Arab School on Science and Technology : Applied Arabic Linguistics and Signal and Information Processing. Rabat. Morocco. 1983 .

[7] - MAKHOUL, J. "Speech Processing" Arab School on Science and Technology: Modern Signal Processing Zabadani - Syria 1983 .

[8] - JAYANT, N.S., and NOLL, P., Digital Coding of Waveforms, Principles and Applications to Speech and Video, Prentice-Hall, New Jerssy.

[9] - FANT G., and al. "O.V.E. II Synthesis Strategy" Proc. of speech Comm. Seminar, Stockholm, 1963 .

[10] - PAILLE J., "Contribution aux Etudes Sur la Synthèse Paramètrique de la Parole. Synthetiseur a formants. Analogue de la source vocale" Thèse Doct. Sciences, Grenoble, 1971 .

[11] - RABINER L.R. "Digital-Formant Synthesizer for Speech Synthesis Studies" J. Acoustic Soc. Am., 43, 822-828, 1968 .

[12] - MRAYATI M., Commande d'un Synthétiseur a Formant par un Ordinateur". Actes des 5 èmes Journées d'Etudes du groupe Communication parlée du G.A.L.F. a Orsay-Paris (Mai 1974) .

[13] - ATAL, B.S., HANAUER S.L., "Speech Analysis and Synthesis by Linear Prediction on the speech wave", J.A.S.A., Vol 50, pp 637-655, 1971 .

[14] - ITAKURA F., SAITO S., "An Analysis-Synthesis Telephony bases on maximum Likelihood Method" Proc. Int. Congr. Acoust., c-5.5, Tokyo, Japan 1968 .

[15] - MARKEL J.D., GRAY A.H., Linear Prediction of Speech, Springer Verlag, New York 1976 .

[16] - MAKHOUL J., A Mixed Source Model for Speech Compression & Synthesis" JASA, Vol 64, No. 6, pp 1577 - 1581, 1978 .

[17] - ATAL B.S., "A New Model of LPC Excitation for Producing Natural Sounding Speech at Low Bit Rates", Int. Conf. ASSP, pp 611-614, Paris 1982.

[18] - DUNN H.K., "The Calculation of Vowel Resonances, and an Electrical Vocal Tract". J.Acoust. Soc. Am., Vol 22, PP 740-753, 1950 .

[19] - STEVENS K.N, KASOWSKI S., FANT G., "An Electrical Analog of the Vocal Tract", J. Acoust. Soc. Am., Vol 25, PP 734-742, 1953 .

[20] - COURBON S.L., "Simulation du conduit Vocal in Technologies Analogiques" 6 èmes Journées d'Etude sur la Parole, Toulouse, 1975 .

[21] - FLANAGAN J.L., ISHIZAKA K., SHIPLEY K.T., "Synthesis of Speech from a Dynamic Model of the Vocal Cords and Vocal Tract", Bell System Tech. Journal, 54, 485-506, 1975 .

[22]- MRAYATI M., GUERIN B."Etude de Caracteristiques des Voyelles Orales Francaises par Simulation du conduit Vocal avec Pertes", Revue d'Acoustique, 9, No. 36, 1976 .

[23]- MRAYATI M., GUERIN B., BOE L. J." Etude de l'Impedance d'Entree du Conduit Vocal; Couplage Source - Conduit Vocal , "Acoustica, 35 Fevrier 1976 .

[24]- MRAYATI M., CARRE R. Relations Entre la Forme du Conduit Vocal et les Caractéristiques Acoustiques des Voyelles Francaises-Etude des Distributions Spatiales, Phonetica 1976 .

[25]- GUERIN B., MRAYATI M., CARRE R. A Voice Source for Formant Synthesis, taking account of coupling with the supraglottal cavities. I.E.E.E. Intern. Conf. On Acoustics, Speech and Signal Processing, Philadelphia, April 1976 .

[26]- MRAYATI M., CARRE R., "Acoustic Aspects of French Nasal Vowels". J. Acoustical Society of America, 57(A), 1975 .

[27]- DESCOUT R., "Speech Synthesis" Arab School on Science and Technology; Applied Arabic Linguistics and Signal and Information Processing, Rabat, 1983 .

[28]- BOE L.J., GUERIN B., MRAYATI M., "La Synthese de la Parole". Image de la Physique. C.N.R.S. Paris PP 80-85, 1974 .

[29]- RABINER L.R., SCHAFER R.W., AND FLANAGAN J.L., "Computer Synthesis of Speech by Concatenation of Formant-Coded Words", Bell Sys. Tech. J., 50, No. 5, 1541-1558, 1971 .

[30]- DIXON N.R, and MAXEY H.D, "Terminal Analog Synthesis of Continuous Speech Using Diphone Method of Segment Assembly". I.E.E.E. Trans. Audio Electro. AU, PP. 40-50 .

[31]- MRAYATI M., SAIT - BONNET M., BOE L.J, "La Synthese Par Diphone, Premier Rapport". Bulletin de l'Institut de Phonetique de Grenoble Volume III, PP. 73-94, 1974.

[32]- HARRIS C.M, "A Study of the Building Blocks in Speech". JASA. No. 25 PP. 962 - 969, 1953 .

[33]- LIBERMAN A.M, et al, "Minimal Rules for Synthesizing Speech". JASA, No. 31, PP. 1490 - 1499, 1959 .

[34]- HOLMES J.N., MATTINGLY I.G., SHEAME J. N., "Speech Synthesis by Rule ". Language and Speech, No. 7, PP. 127-143, 1964.

[35]- CARLSON R., GRANSTROM B., "A Text-to-Speech System based entirely on Rules", I.E.E.E. Int. Conf. ASSP, PP. 686-688. 1976.

[36]- KLATT D.H, "Structure of a Phonological Rule Component for a Synthesis by Rule Program " . I.E.E.E. Trans. ASSP, Vol. 24, PP. 391-398, 1976 .

[37]- RABINER "Speech Synthesis by Rule: An Acoustic Demain Approach" Ph.D.M.I.T. 1967 .

[38]- COKER C., "Synthesis by Rule from Articulatory Parameters", Proc. Conf. on Speech Comm. and Processing, MIT, PP. 55-63, 1967 .

[39]- GUERTI M., "Contibution à la Synthese de la Parole par Diphones en Arabe Standard" MS These, Universite d'Alger, 1983 .

[40]- MOURADI A., NAJIM M., and RAJOUNI A., "On Test to Speech Synthesis in Arabic Language" Arab School on Science and Technology : Applied Arabic Linguistics and Signal and Information Processing. Morocco, 1983 .

[41]- MRAYATI M., MAKHOUL, J. "Man-Machine Communication and the Arabic Language" Arab School on Science and Technology: Applied Arabic Linguistics and Signal Information Processing. Rabat. Morocco, 1983 .

[42]- Al-Waer, m., "Toward a Modern and Realistic Sentential Theory of Basic Structures in Standard Arabic". Ph.D. Thesis, Georgetown University, March 1983 .

[43]- HAYDAR, Y., "Etude de l'Intonation, la Courbe Melodique des Phrases de l'Arabe Standard". Memoire de Maîtrise. Institut de Phonétique de Strasbourg, Juin 1985 .

Automatic Speech Recognition: State of the Art and Applications to Arabic

JEAN-PAUL HATON
CRIN, INRIA
B.P. 239
54506 Vandoeuvre Cedex, France

ABSTRACT

The field of automatic speech processing has been experiencing for the past few years a rapid growth in industrial development. Presently, practical applications are mainly concerned with speech synthesis. However, automatic speech recognition (ASR) is expected to grow during the next ten years and to overcome speech synthesis. This paper is intended to give a comprehensive view of ASR at the level of present industrial applications (isolated and connected word recognition) and also at the level of research problems (multispeaker systems, continuous speech recognition and understanding). In parallel we will mention the experiments carried out for Arabic words recognition and the problems that are to be solved for the understanding of continuous Arabic sentences.

1. INTRODUCTION

The field of automatic speech processing and especially speech recognition has experienced a rapid and important growth during the past few years. Significant advances concern the fields of isolated and connected word recognition, signal processing and acoustic-phonetic mapping, search strategies and system organization.

Several reports in the early seventies presented the work carried out in various countries |1|, |2|, |3|. More recently several papers |4|, |5|, |6| and a book edited by W. Lea |7| are concerned with recent advances in ASR both at the industrial and research levels. This paper presents the work carried out recently by emphasizing the two aspects which presently characterize the field of ASR :

- the development of practical industrial systems,
- the existence of several fundamental problems which still have to be solved in order to implement actual dialog systems, especially at the phonetic and phonological levels.

One of these major unsolved problems deals with the acoustic-phonetic decoding of speech, i.e. the mapping of the continuous acoustic speech wave into a discrete string of phonetic symbols such as phonemes.

2. APPROACHES TO SPEECH RECOGNITION

Basically an ASR system is able to identify and, to a certain extent, to understand words, phrases or sentences pronounced by one or several speakers after a more or less sophisticated training phase. That can be viewed as a classical pattern recognition problem with the usual stages of sensors, preprocessing and decision. However the characteristics of the speech signal and of the speech production process make the ASR problem a unique one for which specific solutions have to be found in close relation to the field of artificial intelligence.

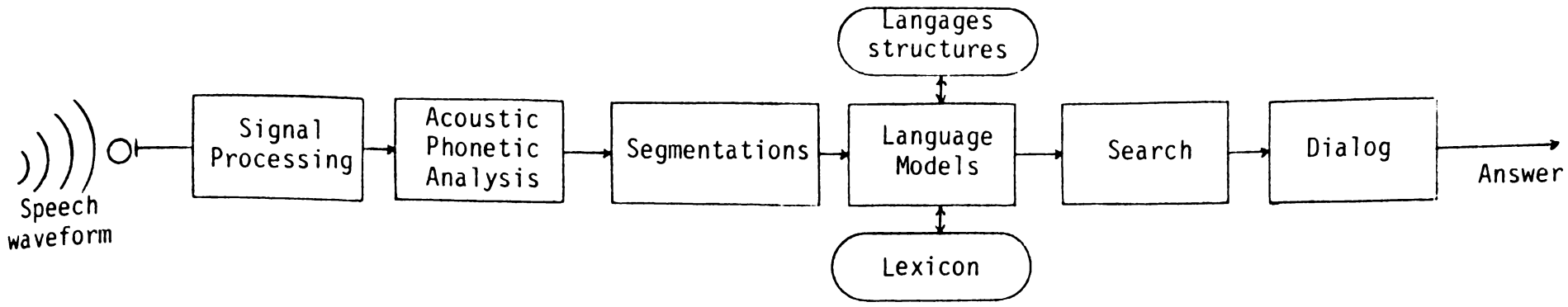
A typical ASR system will have to deal with the following questions : speech waveform coding, signal processing, acoustic-phonetic analysis, different kinds of segmentation, language modeling at various levels, search process, dialog, training procedures.

This is illustrated in figure 1. The linear succession of the various processing levels is just for the sake of simplicity. As a matter of fact these processing levels are highly interrelated in an actual system, and some of them can be merged together.

The difficulty of the ASR task clearly appears when considering the challenges one is faced with in the design of a system. These challenges can be summarized by considering the properties of the speech signal :

- the speech signal is highly encoded : from the brain of a speaker to the speech waveform the latter emits several successive, hierarchical encoding levels are put into action (pragmatic, semantic, syntactic, prosodic, morphologic, phonetic, articulatory, etc.). A speech recognition system will have to do the inverse job in order to decode a spoken input ;
- the speech signal presents a great variability : according to contextual effects different elementary sounds, or phonemes, can look alike whereas the same phoneme can look different, even for the same speaker. Of course the variability is still greater when several speakers are considered ;
- the speech signal is essentially continuous, i.e. the boundaries between words usually do not appear at the acoustic level even though a human listener perceives a discrete sequence of words. Several kinds of segmentation (sentence, phrase, word, syllabe, ...) are carried out by the brain and have to be implemented in an ASR system.

According to the difficulty of the task the field of ASR has been divided into more or less difficult subgoals. Besides the fact that a speech recognizer can be or not be speaker-dependent and that it can handle or not handle syntactic information during the recognition process, the various developments in ASR can be classified according to two orthogonal criteria |8| :



Elements of an automatic speech recognition system
Figure 1

- the kind of speech accepted by the system : isolated words, i.e. words pronounced separetely with an artificial pause between them, or continuous speech ,
- the kind of approach :
  - . global or pattern matching approach in which the utterance of a speaker is considered and processed as a global pattern,
  - . analytical or phonetic approach in which one considers the utterance as constituted by elementary phonic units which are localized and indentified.

Figure 2 shows the various research and development activities in this classification.

Kind of speech / Approach	ISOLATED WORDS	CONTINUOUS SPEECH
GLOBAL	Isolated Word Recognition (small vocabularies)	Word Spotting Connected Word Recognition
ANALYTICAL	Isolated Word Recognition (large vocabularies)	Word Spotting Continuous Speech Recognition and Understanding

Various areas of automatic speech recognition
Figure 2

3. INDUSTRIAL DEVELOPMENTS

Though not as important as in speech synthesis the number of commercially available ASR systems is already significant. About 20 vendors are selling such systems all over the world (USA, Japan, GB, France, ...). The first devices appeared around 1972. The evolution of microprocessors and custom integrated circuits have dramatically decreased the size and the price of these systems. Presently Weitek, a US manufacturer, proposes a single chip at $10 (OEM quantity) for the multi-speaker recognition of small vocabularies (up to 8 words). No doubt that other chips are being developed, especially by large semiconductor manufacturers in the USA and Japan.

Moreover the availability of new fast processing chips from several sources make it possible to implement on silicon sophisticated algorithms including elaborated signal analysis methods such as LPC or cepstrum and continuous speech recognition methods.

Most of the systems presently available are speaker-dependent, isolated word recognition systems. The basic principles involved in such systems will be described in the next section. The performances range from 70% to 99.5% starting with hobbyist peripherals up to sophisticated industrial systems. Industrial systems have been extensively and successfully tested in a number of applications during the past ten years or so, including sorting, parcel routing, assembly line inspection, inventory management, control aids for the handicapped, etc.

Some systems have recently been proposed for the recognition of connected sequences of words, particularly digits. The first system of this kind was deloped in Japan by NEC (DP 100) with the notion of two level dynamic programming (cf. next section). Several tests demonstrate 99% accuracy with speaker training for japanese digit sequences. Verbex also proposes in the USA a high performance multi-speaker, connected word recognition system with accuracy better than 99.5% on digit strings. The cost of this M 1800 system is more than $60K.

The M 1800 system is, to a large extent, speaker independent (average knowledgeable users obtain recognition scores above 96% for isolated digits). Several other companies have recently introduced speaker-independent isolated word systems, including Weitek (already mentioned), NEC and Texas Instruments, since a great need for speaker-independent recognition exists, especially for data entry and retrieval through the telephone.

4. METHODOLOGY FOR WORD RECOGNITION

As we have just seen, the first developments in ASR concerned the recognition of isolated words. The main reason is simplicity : the fact of pronouncing the words separately avoids one of the major problems in speech recognition. Moreover words constitute basic elements of a linguistic system, usually perceived as distinct from each other. Therefore work done in isolated word recognition can be, up to a point, used for identifying words in continuous speech.

We have also seen that most word recognizers - at least for vocabularies below 100 words - use a global method in which the word to be recognized is considered as a global acoustic pattern which is matched against the various reference patterns of the vocabulary.

In that case the problem is typically a pattern recognition problem in which two types of normalization have to be carried out :

- a speaker normalization in order to compensate for inter-speaker variations : we will first get rid of this problem by considering speaker dependent systems and we will then see how this normalization can be done,
- a time normalization in order to compensate for intra-speaker variations in duration and rhythm for a given word. These variations can be important so that a classical pattern matching algorithm cannot be directly applied. Moreover, the rhythm variations are non linear, and a linear normalization will therefore be insufficient.

4.1. Time Normalization

A spoken utterance can be expressed by a time sequence of feature vectors by appropriate feature extraction such as FFT, LPC |9| or filter bank analysis. Let for instance A and B be two utterances :

$$A = (a_k,\ k = 1,\ N)$$

$$B = (b_l,\ l = 1,\ M)$$

where a_k and b_l are feature vectors, for instance $a_k = (a_k^1, a_k^2, \ldots, a_k^p)$ (**p** represents the size of a vector, i.e. the number of features extracted from each speech frame).

The problem of time alignment or time warping between words A and B can be easily visualized in the k-l plane as shown in figure 3 where the feature vectors a_k and b_l are successively displayed respectively along the k-axis and l-axis.

The time warping principle consists of eliminating the time distorsions and of finding the best match between the two time axes. This best match can be described by a mapping function P made up of the sequence of points :

$$P = (p(i),\ i = 1,\ I)$$

where $p(i) = (k(i), l(i))$.

It is assumed that :

$p(1) = (1, 1)$

$p(I) = (N, M)$.

Let **d** be a measure of the "distance" between two feature vectors a_k and b_l :

$d(a_k, b_l) = d(k, l) = ||a_k - b_l||$.

The choice of **d** depends on the nature of the features considered. L1 or L2 distances are often used for filter bank or cepstral coefficients whereas some kind of elaborated statistical distance turns out to be more efficient for LPC coefficients |10|.

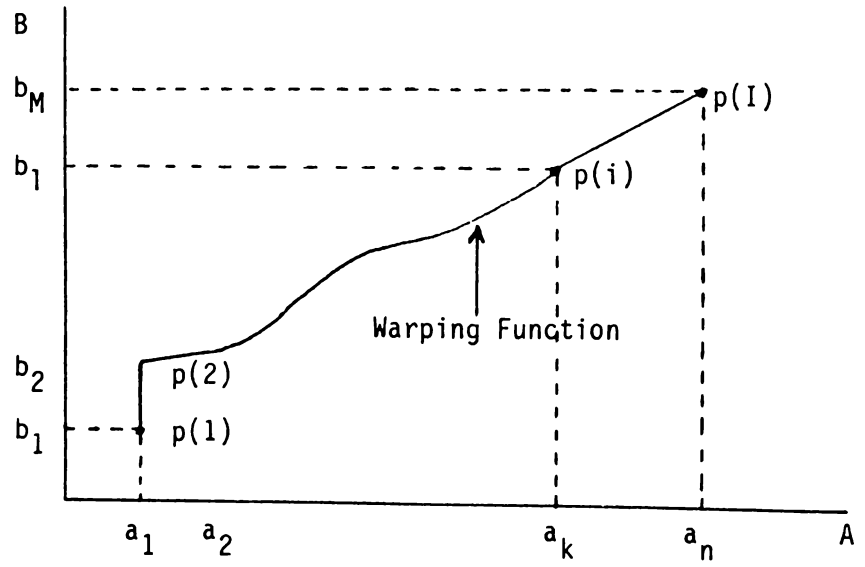
It is now possible to define a weighted summation of distances associated with the path defined by **P** :

$$S(P) : \sum_{i=1}^{I} w(i)\, d(k(i), l(i))$$

where $(w(i))$ is a weighting function.

The time-normalized distance between words A and B is :

$$D(A, B) = \min_P \left[\frac{\sum_{i=1}^{I} w(i)\, d(k(i), l(i))}{\sum_{i=1}^{I} w(i)} \right]$$



Warping Function in A-B Plane
Figure 3

The normalization with respect to **w** is used to compensate for the effect of variable path lengths.

Some restrictions are to be applied to warping function **F** according to specific aspects of speech structures :

- monotony :
  $k(i-1) < k(i)$ and $l(i-1) < l(i)$
- continuity :
  $k(i) - k(i-1) < 1$ and $l(i) - l(i-1) \leq 1$

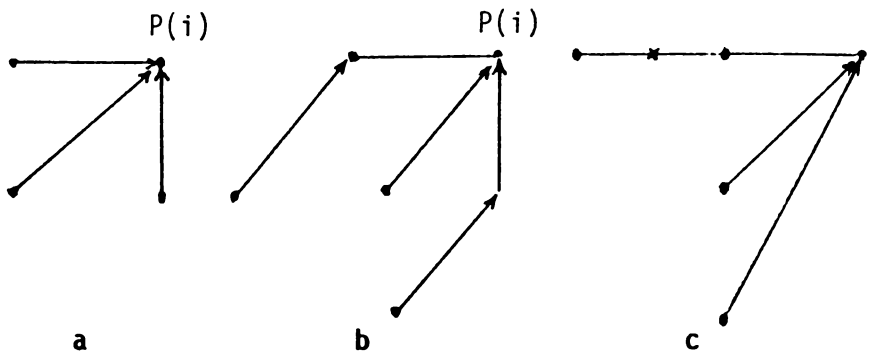
As a result of these two conditions the following relation can be stated between two successive points in a path **P** :

$$P(i-1) = \begin{cases} (k(i),\ l(i)-1) \\ (k(i)-1,\ l(i)-1) \\ (k(i)-1,\ l(i)) \end{cases} \quad \text{with} \quad P(i) = (k(i), l(i))$$

This condition is illustrated in figure 4.a.

- slope condition :

The constraints introduced on the slope of a path **P** are intended to limit unrealistic time-axis warping, like, for instance, between a very short A and a long B.
These constraints result in restrictions on permissible paths : figures 4.b and 4.c show two examples among many others of such constraints, corresponding to slope conditions respectively proposed by Sakoe |11| and Itakura [12]. These constraints have been chosen according to their performance.



Local constraints on warping function in k-1 plane
Figure 4

Several weighting functions can be used. A symetric function [11] defined by :

$$w(i) = (k(i) - k(i-1)) + (l(i) - l(i-1))$$

and such that

$$\sum_{i=1}^{K} w(i) = N + M$$

turns out to give better results in isolated word recognition.

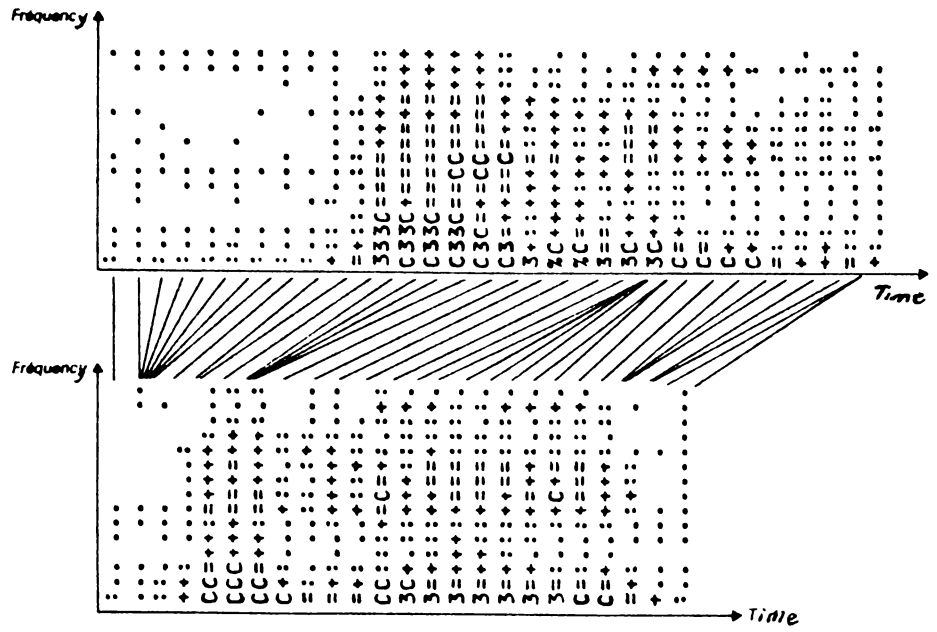
Finally a dynamic programming procedure can be used to solve the minimization problem of the computation of D(A, B). Using the type of constraints described in figure 4.b together with the above mentioned symetric weighting function the following recurrent definition is obtained :

$$D'(k, l) = \min \begin{cases} D'(k-1, l-2) + 2\, d(k, l-1) + d(k, l) \\ D'(k-1, l-1) + 2\, d(k, l) \\ D'(k-2, l-1) + 2\, d(k-1, l) + d(k, l) \end{cases}$$

where **D'(k,l)** represents the accumulated distance along the optimal path from initial stage (1,1) up to state (k,l). Figure 5 illustrates the non linear time warping done by a DP matching algorithm on two occurrences of the Arabic word /Salam/.

The various DP matching algorithms which have been proposed have the advantage of giving a globally optimal solution to the problem of word comparison. However, these algorithms require a large amount of computation time which is usually not compatible with real time requirements on a present mini- or microcomputer for medium range vocabularies (from 50 to 100 words). recent systolic VLSI circuits and pipeline techniques make it possible to design and

build special purpose DP processors |11| for real time operation. Another solution consists of choosing a suboptimal matching algorithm which is much faster and operates quite satisfactorily on isolated word recognition.



Example of DP time warping (adapted after |13|)
Figure 5

The use of DP techniques to compensate for non linear time distorsions began 15 years ago in the USSR |29|, |30|, Japan |23|, France |31| and Great-Britain |36| ; this became very popular around 1978. It is now widely used in almost all isolated word recognition systems. Present studies concern slight improvements in formulation or efficiency, such as the trapezoïdal DP matching |32|, or the use of ordered graph search techniques for reducing the computation time of classical DP algorithms |33|, or else the definition of local constraints which can vary according to local spectral variations of the utterance |34|, |35|.

These techniques can be applied in a straightforward way for the Arabic language since the global recognition of small and medium size vocabularies (less than some hundred words) is language independent. Experiments have been and are still being carried out in several Arabic countries for isolated word recognition. Classical DP matching have proven to be successfull, cf. for instance |58|, |59|. A method based on the restriction of the matching path through a limited number of points of the grid space has also been proposed |60|. These points are selected in accordance with the variations of energy in the speech signal.

4.2. Speaker Independent Systems

In most isolated word recognition performance degrades, either slightly or drammatically according for the robustness of the system, when the speaker is not the one for whom the machine was trained. We have mentioned that the cause of this fact is that the features derived from the acoustic speech wave are highly variable accross speakers. Some kind of normalization is therefore required in order to design multispeaker or speaker-independent systems. There are at least three ways in which this normalization can be done :

- by constructing a set of speaker independent features,
- by adapting the set of reference templates to a particular speaker |15|,
- by statistically characterizing the speaker variability.

Though very attractive, the two first methods need detailed linguistic knowledge that is not presently completely available. The few systems which presently feature some degree of speaker independence use the third method |16|, |17|, |18|, |19|. The way in which the statistical analysis of speaker variability is studied is usually by means of cluster analysis.

A large data base for a population of speakers is first collected and processed ; in this domain an important effort has been made by VERBEX which remains the primary manufacturer in speaker-independent ASR. Then, a clustering procedure is applied to the data in order to obtain representative models of the different types of speech under the form of cluster centers. Some kind of fully automated technique can be used, as the one defined at Bell Labs |17| :

The entire data pool is considered as the initial configuration. Then the algorithm goes as follows :

1) find the minimax center of the current set of words,

2) locate all words whose distance to the center is less than a predetermined threshold,

3) label these words as a cluster and remove them from the pool,

4) repeat until either all words are classified or no more clusters can be found (outliers can be discarded),

5) determine cluster centers either by taking the minimax center of a cluster or by computing an average value of all the words of a cluster. This second solution seems to yield better recognition results.

Few work has been done until now for speaker-independent Arabic recognition |61|. A large effort has to be made in order to collect large data bases of spoken words and the problem is still complicated by the various dialectal forms that can be found.

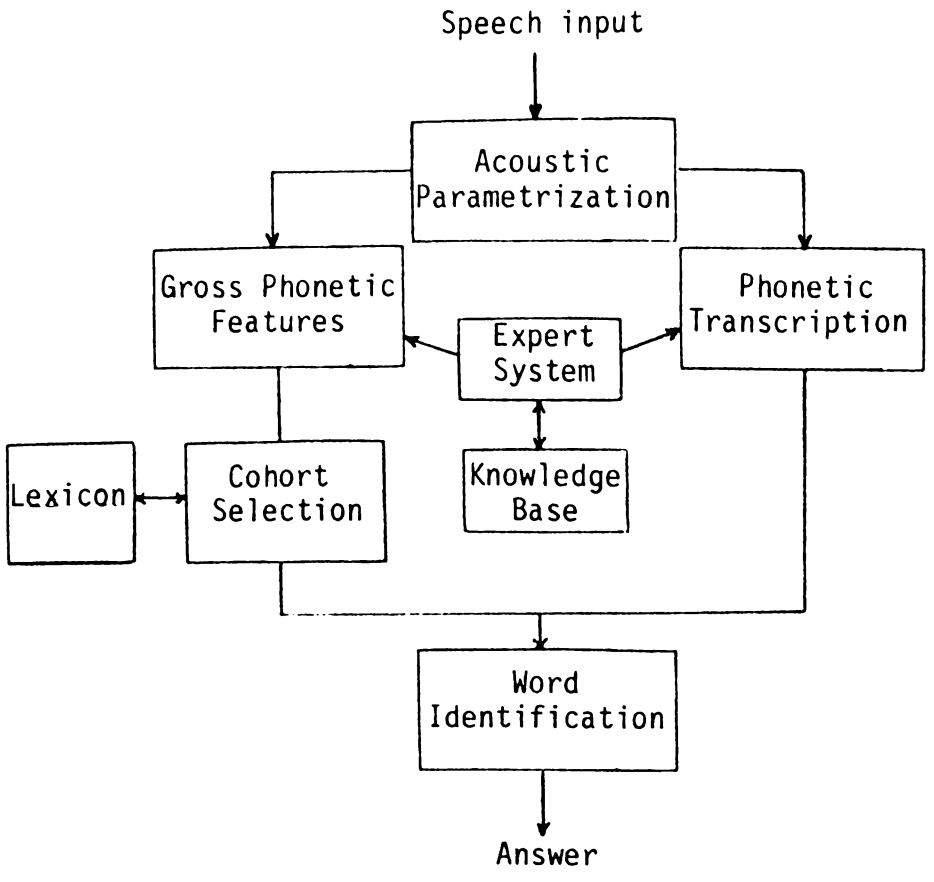
4.3. Large Vocabulary Word Recognition

For large vocabularies (several thousand words) the global techniques we have introduced so far are no longer applicable : DP matching of spectra would not be efficient enough and memory requirements would become enormous. It is therefore necessary to adopt some kind of analytical approach. Reference words in the lexicon are represented by their phonetic transcriptions and the recognition of an unknown word consists of comparing the phonetic transcription of this word (given by an acoustic-phonetic decoder, see section 5) with the transcriptions of the reference words. In order to speed up the recognition it is interesting to select a small subset of reference words (a "cohort") which are phonetically similar to the unknown word on the basis of gross phonetic features and to carry out the comparison only with the words of the cohort. Figure 6 shows the architecture of a system we have developed in Nancy based on this principle. The phonetic and phonological knowledge necessary for carrying out the phonetic analyses is stored in a knowledge base manipulated by an expert system (see the chapter in this book on this subject).

Such systems are mainly language dependent. It is therefore necessary to do extensive studies on the phonetics of Arabic in order to be able to develop such systems. We will see that the expert system formalism can be of great help in order to capture and formalize the knowledge necessary for such tasks.

4.4. Connected Word Recognition

The pronunciation on a word-by-word basis can be unacceptably slow and tedious in a number of practical applications, especially in numerical data entry. The use of connected word recognition systems



Architecture of a large vocabulary word recognition (after |62|)
Figure 6

therefore highly improves the efficiency of vocal inputs. Besides this practical interest connected word recognition is of significant importance in the present state of ASR since the adjunction of syntactic constraints |20| may lead to some key techniques for the general problem of spoken language understanding.

Connected word recognition can be solved by two kinds of methods :
- analytical or semi-analytical methods based on an explicit segmentation of the word string |21|, |22| to be recognized,
- global methods based on whole-word template matching,i.e. methods derived from the isolated word recognition techniques we have just seen.

The extension of DP matching algorithms to this problem has led to the design of very efficient systems using the second approach :
Let $(A_n, n = 1, R)$ be an **R** word vocabulary. The A_n reference word in this vocabulary is made up of a sequence of N_n feature vectors as described previously :

$$A_n = (a_{nk}, k = 1, N_n)$$

Let $B = (c_l, l = 1, M)$ be an unknown input utterance which may be either a single word or a multiple word sequence. The principle of the recognition consists of building a multiple word reference pattern **A** by concatenating isolated patterns A_n, for instance

$$A = a_{n(1)} \oplus A_{n(2)} \oplus A_{n(q)}$$

and of matching **A** against the unknown **B** phrase. The process is repeated, changing the number of words **q** and their order and the **A** string giving minimum distance to **B** is finally selected. Again, the problem can be stated in terms of minimization of a distance :

$$D(B, A_o) = \min_{q,\, n(x)} \left\{ D(B, A_{n(1)} \oplus \dots \oplus A_{n(q)}) \right\}$$

This problem consists of finding the optimal time-registration path between an input utterance and a sequence of isolated templates. This is a generalization of the isolated word recognition problem introduced in the previous section (cf. figure 3).

An exhaustive search would require an enormous amount of computation and cannot therefore be implemented. The use of dynamic programming is again very attractive. Explicit use of such methods was first reported by Sakoe and Chiba |23| in 1971. More recent studies have proposed alternative solutions to the problem :
- a two-level dynamic programming approach (Sakoe |24|) in which the minimization of **D(B, A)** is carried out at the two successive levels of the words and of the phrase. This algorithm is very time consuming and, therefore, has to be implemented on special purpose computers as the one developed by NEC in the DP-100 machine,
- a sampling approach (Rabiner and Schmidt |25|) using the UELM procedure (Unconstrained Endpoint-Local Minimum) for dealing with coarticulation effects. A microprocessor based system also featuring a sub-optimal DP algorithm and a time linearization of utterances has recently been designed by Gauvain |26| with excellent performance, Bridle and Brown |27| also proposed a one-pass algorithm which led to a multi-microprocessor machine commercialized by LOGICA,
- a level building approach (Myers and Rabiner |28|) which has been shown to be equivalent to Sakoe's two level approach but with a sequential aspect which makes it possible to decrease the computation time.

These various **DP** algorithm work fairly well with word strings pronunced at a reasonable rate since they all match these strings against concatened patterns pronunced isolately. They can therefore be directly applied to Arabic word sequences without any adaptation. When the speaking rate becomes larger than the rate of isolated pronunciation then the accuracy of these algorithms tends to decrease significantly. The coarticulation effects are mainly responsible for this effect and a way of improving connected word recognition consists of defining reference patterns embedded in continuous speech instead of pronunced isolately |37|.

5. CONTINUOUS SPEECH RECOGNITION AND UNDERSTANDING

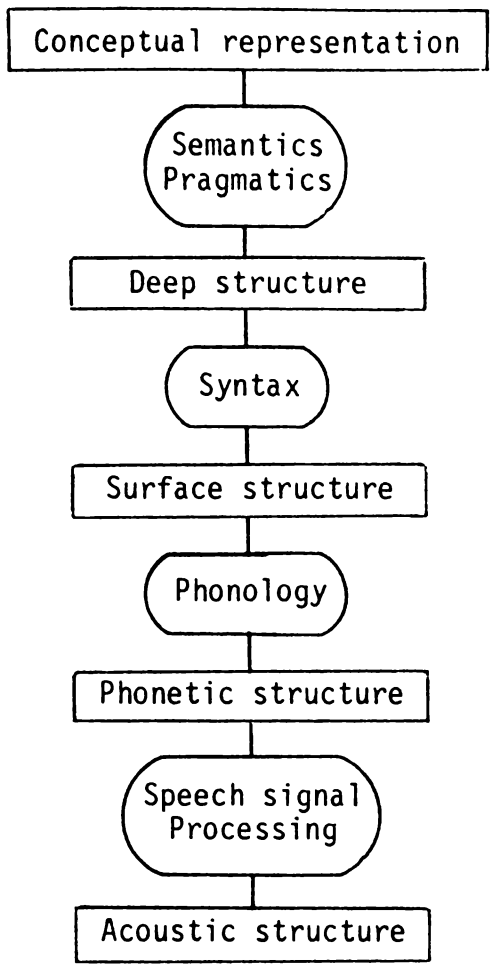
Instead of isolated or connected word recognition continuous speech understanding makes it necessary to use a large amount of knowledge which is to a large extent language dependent. After a brief overview of the problem we will present the main knowledge sources involved in the speech understanding process and discuss some actual system implementations.

5.1 Overview of the Problem

We have already mentioned the difficulty of the task which consists of recognizing continuous speech. However the improvement in voice data entry compared to isolated words (naturalness, ease, speed,...) makes it worthwhile to keep on making an important research effort in this domain. The five year ARPA Speech Understanding Project (1971-1976) had a very benefical effect not only within the USA but also all around the world. The distinction between speech recognition and speech understanding was proposed at that time |38| : continuous speech recognition tries to identify every element of a spoken message whereas in speech understanding one aims at capturing the meaning of a message even though all its elements are not identified. That implies the use of high level linguistic knowledge in order to compensate for the uncertainty of the acoustic-phonetic level. As a matter of fact the necessity of linguistic information was already known as early as the 1960s |39|, but the systems developed during the ARPA Project were among the first to extensively use such information.

Even if a machine has not to mimic the mechanism of a human listener's brain it is important to have

some kind of model of speech perception as a guide for continuous speech recognition studied. Figure 7 shows a possible model in which appears the hierarchical organization of processing levels. Several structures can be defined from the acoustic structure to the conceptual representation. These structures will appear more or less explicity in automatic recognition systems since the different processing levels involved in this model correspond to knowledge sources which will be used in such systems, i.e. syntax, semantics and pragmatics.



A model for human speech perception (after [40])
Figure 7

Syntax provides the most immediate constraints on word sequences. The role of syntactic knowledge is twofold :
- determine whether a particular sequence of words can belong to the processed language or not,
- predict the words which can occur at a given place within a sentence. This predicting capability of syntax is of particular importance in ASR.

It is generally admitted that the usual order for Arabic sentences is Verb-Subject-Object, with a large number of possible variations order. This has to be taken into account in order to build parsers for spoken Arabic. Models developed for written language |63| can be partly reused but, as for other languages, they have to be adapted to the specificity of spoken language.

Semantics can determine if a syntactically correct sentence is meaningful. Moreover semantic information can also be used in order to predict sentence constituents (words or phrases) on a meaningful basis.

Pragmatics enables us to determine whether a meaningful sentence is plausible according to the context of the ongoing dialog. It is obvious that this kind of information can also be used in a predictive way. It will also provide a framework for man-machine dialog control.

A number of interesting studies have been carried out concerning the use of separate representations for syntax and semantics |41|, |42|, |43|, especially with the formalism of context-free grammars |44|. However the parsing of a spoken sentence is far more difficult than for formal languages, particularly due to the fact that the input (i.e. a phoneme string or lattice) is highly undeterministic. The separation between syntax and semantics is acceptable for artificial, highly constrained languages for which a description in terms of a set of context-free rewriting rules is workable. For natural or pseudo-natural languages it is no longer possible to keep this distinction. Presently, most systems attempt to integrate syntax and semantics in a unified model. This is usually done by using different kinds of syntactic representation, for instance networks, augmented in various ways to account for contextual phenomena.

Some of these solutions will be described later on in this paper. In all cases, the importance of the acoustic-phonetic front-end in a system is obvious since the linguistic levels work on the data given by the phonetic transcription. Sophisticated linguistic processing cannot compensate for a poor phonetic input and therefore an important effort still has be done at this level |7|.

5.2. Acoustic-Phonetic Decoding

The role of this processing level is to transform the continuous acoustic data into discrete phonetic units such as syllables, diphones, phonemes or phones. This transformation cannot be carried out by straightforward signal processing techniques : it is necessary to rely on the underlying speech production process and on the knowledge associated with a given language. The phonetic decoding is therefore for its major part language dependent although general models and approaches can be reutilized from one language to another.

The phonetics of Arabic has been studied for a very long time |65| but a great amount of work has to be done in speech recognition since very few studies have been until now oriented towards this direction. The problem is complicated in Arabic by the fact that there exist many variations from Modern Standard Arabic, according to the country |66|.

The Arabic language presents specific features that must be studied from an automatic recognition viewpoint. It is for instance characterized by a relatively large number of back consonants, especially pharyngeals, uvulars and pharyngealized. The acoustic-phonetic analysis of such consonants has to be studied in details since these sounds are not common in languages like French or English.

The automatic phonetic decoding of speech makes it necessary to interpret the acoustic data by a contextual reasoning taking into account all available knowledge. An important knowledge source is phonology since it is related to the alterations of sounds in context (liaisons, co-articulations, assimilations, etc.) which are for the most part language dependent. Original contextual effects appear in Arabic, particularly due to the coarticulation effect of emphasis induced by pharyngealized consonants into the neighbouring sounds. This effect is limited to a syllable but it can extend outside the syllable up to the entire word in some dialectal forms of Arabic. This phenomenon is not totally well known and must be studied on large amount of data in order to extract a set of contextual rules. The technology of expert systems may be of great help for that since it provides a framework for capturing and formalizing the knowledge of human expert phoneticians |67|.

Suprasegmental phenomena are important as well. They are related to prosody, i.e. stress, duration, etc. All Arabic phonemes have short and long (or geminated for consonants) realizations. These phenomena directly interact with the phonetic level

through the notions of stressed or unstressed vowels, vowel reduction, tonic accents, etc. They have still to be extensively studied, once again on large amount of data in close interaction with phoneticians and linguists. More generally, prosody is a very important knowledge source for language understanding but it does not exist any satisfactory operational model that makes it possible to use prosody in an automatic system. This is true for English or French as well as for Arabic.

5.3. System Organization

The design of continuous speech recognition systems can be considered from three complementary points of view :
- the speech wave viewpoint for which the acoustic speech signal is just a signal among others. Recognition simply consists of applying various mathematical signal processing techniques derived from statistics, Fourier transform, etc.,
- the speech production viewpoint refers to a model of the human vocal system in order to derive recognition parameters such as vocal tract resonances, coarticulation movements, articulatory parameters, vocal cord vibration, etc.,
- the speech perception viewpoint suggests we take into account the knowledge about the human brain mechanism of speech perception and use recognition parameters and schemes which have been shown experimentially to have some importance at that level.

Of course no one of these viewpoints can lead alone to workable speech recognizers. Actual systems combine to some extent the three approaches and give an emphasis to a particular one, according to the general philosophy of the system. Therefore, the organization of any ASR system will reflect this distinction between the three basic viewpoints.

The understanding of a sentence implies, in fact, the cooperation of various knowledge sources (KS) (phonetics, phonology, prosody, lexicon, syntax, semantics, pragmatics, etc.) which can be very different and have to be activated at the right moment when certain conditions are verified.

A KS scheduler is in charge of assigning priorities between the KSs which want, at the same time, access to the sentence representation and modify it and, therefore, controls the interaction between the KSs. The architecture we have just described is rather general in Artificial Intelligence and can be applied to other understanding systems in which the problem consists of intepreting complex "objects" (speech, but also images, physical signals, etc.). Simply stated there are two general models of KSs interaction (cf. the chapter on expert systems for more details) :
- the hierarchical model which is the most straight forward one : an explicit hierarchy is predefined between the KSs, from the acoustic level to the conceptual level. Interactions are only available between one level and the next higher ont. This is of course very simple to implement but very limitative. An important improvement consists of allowing both feed forward and feed back interactions and also interactions between non adjacent levels under the control of a supervisor. This idea corresponds to several systems such as HWIM |45| from BBN, LITHAN |46| from Kyoto University, MYRTILLE II |47| from CRIN, etc.
- the heterarchical model which allows any KS to interact with any other one. A modified version of this model is particularly attractive, i.e. the blackboard model. In this model each KS is independent and communicates only through a general, multidimensional data-base called blackboard and made up of hypotheses. Whenever its preconditions are fulfilled, any KS can access the database and modify, create or complete any hypothesis. The model is therefore data-driven and does not need any supervisor. It is well-known in Artificial Intelligence and has been implemented in the HEARSAY II |48| speech recognition system at Carnegie Mellon University. Even though the blackboard model represents a simplification over the general heterarchical model the control of search remains complex and time-consuming.

It can be noticed that, instead of trying to implement some kind of sophisticated interaction between very different knowledge sources, one solution consists of integrating all of them into a single, precompiled network. This network gives a complete description of all possible sentences of the language ; the processing of a sentence is reduced to the search for the best path through the network. In the HARPY system |15|, also from CMU, which uses this model, this search is carried out by a beam search algorithm. HARPY turned out to be the most efficient system designed in the framework of the ARPA project and the only one meeting most of the requirements described by this project. However the size of the network increases rapidly with the complexity of the language and the solution becomes untractable for pseudo-natural language processing.

In any model, the level of the words, corresponding to the lexical knowledge source, is of primary importance. We have already mentioned the linguistic importance of the words in a language. In terms of automatic processing the words are situated at a "key position" since they can be accessed either in a bottom-up way from the signal level and then the phoneme level or in a top-down way from the language structures.

Therefore, the different techniques involved in the representation and use of a lexicon have been extensively studied |49|. The use of a lexicon includes word spotting, verification and hypothesization. These different operations are usually performed on a string or lattice of phonemic symbols representing the sentence pronounced by a speaker.

Word spotting consists of locating a given word in a continuous stream of speech or its phonetic transcription. It can in a certain sense be considered as a particular case of the word verification problem in which one tries to verify whether a word hypothesized by other KSs can be found in a phoneme lattice. A lot of methods have been proposed for solving these two problems, some of them use extended versions of dynamic programming or of a stochastic version, i.e. the Viterbi algorithm |50|.

Word hypothetization can be carried out either in a top-down or a bottom-up way. Top-down hypothesizing is rather straight-forward : according to the part of the sentence already processed and to the knowledge about syntax, semantics and vocabulary certain subsets of words are only permissible. The bottom-up operation is more complicated, especially when the size of the vocabulary increases. In fact it is necessary to combine both operations in order to increase the efficiency of word hypothesization.

Syllables have been shown to be interesting units at the lexical level. They have been used extensively for instance among others in the ARIAL Project |51|, in the NOAH word hypothesizer |52| or in the systems developed in Torino by De Mori |53|. NOAH uses a wordsyllable tree representation and can be considered as one of the most powerful hypothesizer for very large vocabularies (over 1000 words).

The Arabic language presents original properties for the derivation of words from roots and basic lexical forms |64|. Rules and model governing this process have to be investigated in the framework

of speech recognition in order to speed up the lexical processing.

5.4. Control and Search Strategies

The errors and ambiguities which appear in the automatic decoding of the speech signal together with the limited capacity of human knowledge modeling make it necessary to design strategies for the search for the best (in some predefined sense) solution among all possible solutions. This is again a central problem in the field of Artificial Intelligence and the solutions found in speech recognition are often derived from general AI techniques.

Usual exhaustive blind search methods, depthfirst and breadth-first, are unefficient and it is therefore necessary to use some heuristics or mathematical methods in order to improve the search efficiency. A bestfirst search is often used : it consists of keeping at each time the most likely solution and to backtrack when necessary. This strategy is simple to implement but it can only be used for limited, artificial languages. Another interesting method consists of pruning the search tree, keeping only some of the best solutions and processing these solutions in parallel. This method was implemented in several systems, for instance HARPY already mentioned, LITHAN, KEAL |54|.

Another interesting idea is the so-called island driven strategy which consists of processing a sentence from one or several arbitrary starting points in the utterance. These starting points are chosen as reliable anchor-points which will facilitate the processing of the remaining part of the sentence |55|.

An interesting approach, very different from the preceding ones, has been introduced by Baker |56| and Jelinek. In the system developed by Jelinek et al. at IBM |57| the language is represented as a Markov process under the form of a stochastic transition network. The major part of the system is a linguistic decoder the role of which is to determine the most probable sentence according to the phone string transcribed from the speech input. This model extensively uses conditional probabilities of linguistic elements (phonemes, diphones, words). In a recent work, Jelinek investigates the recognition of sentences with very large vocabularies (several thousand words) on a word-by-word pronunciation basis.

Clearly, no one of the above mentioned methods is completely satisfactory and no one completely solves the problem of speech recognition control. Important work has still to be done. Results obtained in Artificiel Intelligence and other related fields could be of great importance. For instance expert systems could provide a very promising framework for solving some subproblems of ASR. Great importance has also to be devoted to the work carried out in psycho-linguistics, cognitive sciences, and in the design of graceful man-machine dialog and interaction.

CONCLUSION

In this paper we have presented an overview of automatic speech recognition from isolated word recognition to continuous speech understanding. This field presents a typical example of the mixing of Pattern Recognition and Artificial Intelligence techniques, from the level of signal processing and speech unit identification up to the level of sentence understanding.

Several industrial developements exist, particularly in isolated and connected word recognition but fundamental problems still have to be solved before being able to implement intelligent man-machine dialog systems. Active research in Artificial Intelligence, phonetics, cognitive sciences and forthcoming powerful VLSI chips make us hope that such systems will appear within the end of this decade. Some interesting experiments have been carried out for the recognition of Arabic words using general techniques such as dynamic programming. The recognition/understanding of continuous sentences in Arabic is only beginning. It makes it necessary to collect a large amount of knowledge related to the different aspects of the problem by studying important set of speech data in cooperation with phoneticians and linguists.

REFERENCES

1. Hill D.R., Man-Machine Interaction Using Speech, in Advances in Computers, F.L. Alt et al. eds, Academic Press, 1971.

2. Hyde S.R., Automatic Speech Recognition : Literature, Survey and Discussion, in Human Communication : an unified view, E.E. Davis, P.B. Denes eds, Mc Graw Hill, 1972.

3. Haton J.P., Reconnaissance automatique de la parole : bilan de vingt années de recherche et tendances actuelles, Ann. Telecom., 27, n° 3-4, pp. 77-88, 1972.

4. Martin T.B., Practical Applications of Voice Input to Machines, Proc. IEEE, 64, n° 4, pp. 487-501, 1976.

5. De Mori R., Recent Advances in Automatic Speech Recognition, Int. Conf. Pattern Recognition, Kyoto, Japan, November 1978.

6. Doddington G.R., Schalk T.B., Speech Recognition : Turning Theory to Practice, IEEE Spectrum, 18, n° 9, pp. 26-32, 1981.

7. Trends in Speech Recognition, W.A. Lea editor, Prentice Hall, 1980.

8. Haton J.P., Reconnaissance de la parole : l'état des recherches, 01 Informatique, n° 131, pp. 44-51, 1979.

9. Makhoul J., Linear Prediction : a Tutorial Review, Proc. IEEE, 63, pp. 561-580, April 1975.

10. Itakura F., Minimum Prediction Residual Applied to Speech Recognition, IEEE Trans. ASSP, 23, pp. 67-72, February 1975.

11. Sakoe H., Chiba S., Dynamic Programming Algorithm Optimization for Spoken Word Recognition, IEEE Trans. ASSP, 26, ,° 1, pp. 43-49, 1978.

12. Myers C. et al., Performance Tradeoffs in Dynamic Time Warping Algorithms, IEEE ASSP, 28, n°6, 1980.

13. Haton J.P., New Trends in Communication with Robots, 2d Internat. Meeting on Artificial Intelligence, Leningrad, 1980.

14. Haton J.P., Contribution to Automatic Speech Analysis and Recognition, Doctorat d'Etat Thesis, University of Nancy, January 1974 (in French).

15. Lowerre B., The HARPY Speech Recognition System, Ph. D. Thesis, Carnegie-Mellon University, Pittsburgh, USA, 1976.

16. Rabiner L.R. et al., Speaker Independent Recognition of Isolated Words Using Clustering Techniques, IEEE Trans. ASSP, 27, n° 4, pp. 336-349, 1979.

17. Rabiner L.R., Wilpon J.G., Speaker Independent Isolated Word Recognition for a Moderate Size (54 words) Vocabulary, IEEE Trans. ASSP, 27, n° 6, pp. 583-587, 1979.

18. Baker J.M., How to Achieve Recognition : a Tutorial/Status Report on Automatic Speech Recognition, Speech Technology, 1, n° 1, pp. 30-43, 1981.

19. Levinson S.E., Speaker Independent Connected Word Recognition, in Automatic Analysis and Recognition of Speech, J.P. Haton editor, Reidel, 1982.

20. Levinson S.E., Rosenberg A.E., A New System for Continuous Speech Recognition, Preliminary Results, Proc. ICASSP-79, pp. 239-243, 1979.

21. Haton J.P., Morel O., Automatic Recognition of Connected Digit Sequences by Means of Segmentation and Dynamic Programming, Proc. IEEE ICASSP, Washington, pp. 245-248, 1979.

22. Zelinski and Class, A Segmentation Algorithm for Connected Word Recognition Based on Estimation Principles, Proposed to IEEE Trans. ASSP.

23. Sakoe H., Chiba S., A Dynamic Programming Approach to Continuous Speech Recognition, Proc. Int. Congress on Acoustics, Budapest, 1971.

24. Sakoe H., Two-Level DP-Matching. A Dynamic Programming Approach for Connected Word Recognition, IEEE Trans. ASSP, 27, n° 6, pp. 588-595, 1979.

25. Rabiner L.R., Schmidt C.E., Application of Dynamic Time Warping to Connected Digit Recognition, IEEE Trans. ASSP, 28, n° 4, pp. 377-388, 1980.

26. Gauvain J.L., Mariani J., A Method for Connected Word Recognition and Word Spotting on a Microprocessor, Proc. ICASSP-82, Paris, 1982.

27. Bridle J.S., Brown M.D., Connected Word Recognition Using Whole-Word Templates, Proc. Inst. of Acoustics, Automn Conf., 1979.

28. Myers C.S., Rabiner L.R., A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition, IEEE Trans. ASSP, 28, n° 2, 1981.

29. Velichko V.M., Zagorujko N.G., Automatic Recognition of 200 Words, Int. J. Man-Machine Studies, 2, p. 223, 1970.

30. Slusker G.S., Nelinejnyj Method Analiza Recevych Signalov, trudy NIIR, n° 2, pp. 76-82, 1968.

31. Haton J.P., The Use of Dynamic Algorithm in Automatic Speech Recognition, 7th Int. Conf. Cybernetics, Namur, 1973.

32 Okochi M., Sakai T., Trapezoïdal DP Matching with Time Reversibility, Proc. ICASSP-82, pp. 1239-1242, 1982.

33. Brown M.K., Rabiner L.R., Dynamic Time Warping for Isolated Word Recognition based on Ordered Graph Search Techniques, Proc. ICASSPP-82, pp. 1255-1258, 1982.

34. Silverman H.F., Dixon N.R., State Constrained Dynamic Programming (SCDP) for Discrete Utterance Recognition, Proc. ICASSP-80, pp. 169-171, 1980.

35. Moore R.K. et al., Locally Constrained Dynamic Programming in Automatic Speech Recognition, Proc. ICASSP-82, pp. 1270-1273, 1982.

36. Warren J.H., A Dynamic Pattern Matching Algorithm with Applications to Automatic Speech Recognition, Int. Conf. on Machine Perception of Patterns and Pictures, Teddington, GB, pp. 141-150, 1972.

37. Rabiner L.R. et al., An Embedded Word Training Procedure for Connected Digit Recognition", Proc. ICASSP-82, pp. 1621-1624, 1982.

38. Newell A. et al., Speech Understanding Systems : Final Report of a Study Group, 1971.

39. Lindgren N., Machine Recognition of Human Language, IEEE Spectrum, n° 2, March, April, May Issues, 1965.

40. Liberman A.M., The Grammars of Speech and Language, Cognitive Psychology, 1, pp. 301-323, 1970.

41. Woods W.A. et al., Speech Understanding Systems, Final Technical Progress Report, volumes I-V, B.B.N., Cambridge, USA, 1976.

42. Erman L.D., An Environment and System for Machine Understanding of Connected Speech, Ph. D. Thesis, Carnegie-Mellon University, 1974.

43. Haton J.P., Pierrel J.M., Organization and Operation of a Connected Speech Understanding System at Lexical, Syntactic and Semantic Levels, Proc. IEEE ICASSP, Philadelphia, 1976.

44. Chomsky N., Syntactic Structures, Mouton, 1957.

45. Wolf J.J., HWIM : A Natural Language Speech Understander, Proc. IEEE Conf. on Decision and Control, New Orleans, 1977.

46. Sakai T., Nakagawa S., Speech Understanding System LITHAN and some Applications, 3d IJCPR, Coronado, 1976.

47. Pierrel J.M., Haton J.P., Data-Structures and Architecture of the MYRTILLE II Speech Understanding System, 4th IJCPR, Kyoto, 1978.

48. Erman L.D., Lesser V.R., The HEARSAY II Speech Understanding System : A Tutorial, in Trends in Speech Recognition, W.A. Lea editor, Prentice Hall, 1980.

49. Haton J.P., The Representation and Use of a Lexicon in Automatic Speech Recognition and Understanding, in Spoken Language Generation and Understanding, J.C. Simon ed, D. Reidel, 1980.

50. Forney G., The Viterbi Algorithm, Proc. IEEE, 61, n° 3, pp. 268-278, 1973.

51. Pérennou G., The ARIAL II Speech Recognition System, in Automatic Analysis and Recognition of Speech, J.P. Haton editor, D. Reidel, 1982.

52. Smith A.R., Sambur M.R., Hypothesizing and Verifying Word for Speech Recognition, in Trends in Speech Recognition, W.A. Lea editor, Prentice Hall, 1980.

53. Demichelis P., De Mori R., Laface P., Interaction between Auditory, Syllabic and Lexical Knowledge in a Speech Understanding System, in Automatic Analysis and Recognition of Speech, J.P. Haton editor, D. Reidel, 1982.

54. Mercier G. et al., The KEAL Speech Understanding System, in Spoken Language Generation and Understanding, J.C. Simon editor, D. Reidel, 1980.

55. Haton J.P., Mohr R., A New Parsing Algorithm for Complex Patterns and its Applications, 3d IJCPR, Coronado, 1976.

56. Baker J.K., The DRAGON System : an Overview, IEEE Trans. ASSP, 23, n° 1, pp. 24-29, 1975.

57. Jelinek F., Self Organized Speech Recognizers, in Automatic Speech Analysis and Recognition, J.P. Haton editor, D. Reidel, 1982.

58. Rehif N., Zouabi B., Système de reconnaissance globale de mots isolés, Arab School on Science and Technology : Applied Arabic Linguistics and Signal Information Processing, Rabat, Morocco, 1983.

59. El Meliani R., Reconnaissance globale de mots arabes, Thesis, Faculty of Sciences, Rabat, Morocco, 1985.

60. Hashish M.A., Elkheshen A.T. and Elghonemy M.R., Experiences in Isolated Arabic Word Recognition, Worshop on Computer Processing of the Arabic Language, Kuwait, 14-16 April, 1985.

61. Elghonemy M.R., Speaker Independent Isolated Arabic Word Recognition System, Ph. D. Thesis, Caire University, 1985.

62. Mari J.F. and Haton J.P., Some Experiments in a Thousand Word Vocabulary Recognition System, Proc. ICASSP, San Diego, 1984.

63. Fassi Fehri A., Une syntaxe réaliste de l'Arabe, ASST : Applied Arabic Linguistics and Signal Information Processing, Rabat, Morocco, 1983.

64. Moutaouakil A., La dérivation lexicale en arabe : racines et schèmes, ASST : Applied Arabic Linguistics and Signal Information Processing, Rabat, Morocco, 1983.

65. Hadj Salah A., Linguistique et phonétique arabes, ASST : Applied Arabic Linguistics and Signal Information Processing, Rabat, Morocco, 1983.

66. Ghazali S., Elements of Arabic Phonetics, ASST : Applied Arabic Linguistics and Signal Information Processing, Rabat, Morocco, 1983.

67. Haton J.P., Knowledge-Based and Expert Systems : Basic Principles and Application to Language Understanding, in this volume.

ARABIC SOFTWARE

Arabization of Software Tools

AYMAN S. ABU-MOSTAFA and SAMIR A. ARAFEH
Arabic Software & Computers, Inc.
725 Deep Valley Drive
Rollings Hills Estates, California 90274, USA

ABSTRACT

Software tools are classified and their arabization is discussed. Total arabization is defined and contrasted with partial arabization techniques referred to as "dubbing". Standardization of Arabic codes and character sets are emphasized. Problems encountered during the arabization process are listed and successful arabized software is highlighted. Examples from existing arabized software are given.

I. INTRODUCTION

Software tools may be classified into the following categories:

1. Operating systems.
2. Program translators or programming languages, i.e., interpreters, compilers and assemblers.
3. System utilities.
4. Production tools such as word processors, database management systems, forms and report generators, and spreadsheets.

With the aid of these tools, computer programmers and users are able to program their computer systems to perform all kinds of services and activities ranging from entertainment, education and training to business, commercial and scientific applications.

One approach to software tool arabization is to enable the user to enter and retrieve information in Arabic while the system operates in its original language. This is usually achieved by adding an Arabic or bilingual keyboard and a hardware component to generate Arabic characters. Similar results can be achieved by software alone using the graphics abilities of the computer. Systems using this approach are generally called "converted systems"(1). The majority of the Arabic computer systems on the market today are converted systems, e.g. AraApple, ZX-Spectrum, MSX-Sakhr (Yamaha), and Arabized versions of IBM, HP, and NCR personal computers. Such systems have the advantages of ease of production and lower cost. This approach is referred to as "dubbing"(1) since the operating system and the programming remain in the original language. A prerequisite for programmers, operators, and advanced users of these systems is that they should know the original language.

At the mini and main-frame computer level, the arabization process is either limited to the arabized terminals connected to the system with its native operating system, or modified operating system that accomodates the arabization requirments. To date, these types of computers are still using the "dubbing" approach.

Total arabization, on the other hand, should permit the user to interact with and program the computer entirely in Arabic without ever needing to know the other language. Such approach can only be implemented by arabizing the majority of the software tools mentioned above; in particular the operating system and the program translators.

There are a number of problems which always arise during arabization attempts. Some of these problems are:

1. The Arabic character set: A typical problem is whether to regard the "hamza" as a separate letter or as a subcharacter and how to handle its different positions. Another problem is whether to think of the "lamm-alef" as one character or a multiple character.

2. The diacritics or "harakat": Whether they are needed is a debatable issue. They pose a special problem when combined with other diacritics, such as the "shadda", and the combined diacritic is to be superimposed on a character during display and printing.

3. Proportional spacing: The sizes of Arabic characters are not the same. The terminal "seen" and "saad" letters, for instance, occupy two display cells even though each is a single letter whereas the "lamm-alef" fits perfectly in one display cell despite being two letters. The problem becomes more challenging in printing since printed character sizes vary tremendously in order to produce quality printouts.

4. Standard syntax: When translating English-like syntax of a programming language, there are no standards established to enable the translation process to adhere to a unified and agreed-upon set of Arabic synonyms.

These are but a few of the problems that need to be dealt with in the arabization process. In the next sections, we will discuss the particulars of arabizing each category of software tools.

II. ARABIZATION OF OPERATING SYSTEMS

Operating systems exist to do a variety of program and data management duties. Any operating system should be able to load and execute programs, accept and process input and output, access and manage memory and peripherals and respond to users' queries. Again, operating systems may be dubbed to allow Arabic input/output, or they can be extended further to allow command processing in Arabic. The arabized operating system may be written in a foreign assembly language. The Alraed operating system, for instance, is written in Intel's 8080 processor code. However, total arabization covers program translators which include the assembler.

The portion of the operating system which requires the most attention during arabization is the input/output segment. Upon pressing a key, this segment should check for the "language mode" (for bilingual systems) and perform the necessary functions to generate the proper character for display. The contextual analyzer, which determines the correct character shape by examining its location in the word, can be a part of this segment or it can be stored in ROM.

The Arabic Character Set:

An essential step towards arabization of any software tool is the decision on the Arabic character set. To start with, it should be decided whether 7-bit coding is sufficient or full 8-bit coding is necessary. 7-bit coding allows faster and more reliable character transfer. Except for Arabic-only systems this approach causes problems for bilingual processing since the two languages

are using the same ASCII codes. This necessitates the need to introduce an additional switching flag to identify the language. The most famous 7-bit Arabic codes are ASMO-449 (2) (which is based on the original CODAR-U standards introduced in the late 1970's) and ARCII which was introduced by Arabic-Latin Information Systems, Inc. (3). Figures (1) and (2) show these two codes. Full 8-bit coding, on the other hand, facilitates bilingual processing and leaves room to allocate codes to function keys. Examples of such codes are the RCTC (Research Computer Technology Corporation) extended ASCII code and the IBM 3270 8-bit code. Figure (3) shows the RCTC code. There is no unanimous agreement on which coding method to use. In fact, there is little agreement on what an Arabic character set should include. ARCII code, for example, includes four Farsi characters and four shapes for the "lamm-alef" combination as shown in Figure (2) and the RCTC code has ten different codes for the "hamza" as shown in Figure (3). The need for standardization is definite. The final character set should not disregard the "hamza" and the diacritics which are often necessary for correct reading. Several suggestions for standardization are found in the literature. See, for instance, references 4-7.

Hardware Considerations:

There is very little progress in hardware technology specifically targeted for Arabic processing. Most efforts have been in creating firmware that is designed to ease the burden on the software in areas like contextual analysis and character generation(8). Probably all microprocessors used in Arabic computers today are ready-made. There is a little need to modify these processors. Using existing microprocessors makes it possible to run thousands of useful software programs available in other languages. This means that it is better to arabize an existing operating system rather than write a new one from scratch. The most famous arabized operating systems for microcomputers are Alraed of RCTC, Khalid of Saudi Computer Industries, Inc.,(9) and ARABSYS of Gulf Data, Inc. Table I shows the commands of the popular operating system CP/M and their arabization in Alraed operating system. Other efforts in dubbing existing operating systems include BCON which was recently introduced by Arabic-Latin Information Systems, Inc. (ALIS) which arabizes MS-DOS.

III. ARABIZATION OF PROGRAM TRANSLATORS

Program translators are software programs themselves. They translate a source program written in a high-level programming language into the low-level machine language of the particular microprocessor. They include compilers, interpreters and assemblers. Arabization of program translators, therefore, means arabization of the commands, statements and functions of the high-level language and adjusting the program translator accordingly. The assembly language in which the program translator is usually written may not be arabized. There are essentially three approaches to arabization of program translators:

(A) Source Language Conversion:

The first approach is using a "language converter". In this approach, the Arabic source program is converted, using the language converter, to the foreign programming language which the original program translator understands. This creates

translated to machine language by the original program translator. For example, a source program written in Arabic BASIC is first converted to English BASIC, then the resulting BASIC program is run through the BASIC interpreter. The program translator, the BASIC interpreter in this example, is not altered. This procedure may slow down program development since there is a need to convert the source program every time the program is modified. This approach, however,

TABLE I
ARABIZED CP/M
OPERATING SYSTEM COMMANDS

English Command	Arabic Command	Function
ARAB	-	Switch to Arabic commands
COPY	انسخ	Copy files
DIR	فهرس	Display the directory
ERA	ازل	Erase a file
FORMAT	القرص	Format a disk
REN	سمي	Rename a file
SAVE	احفظ	Save memory on disk
STAT	مرتب	Display disk space
SYSGEN	اعتماد	Copy CP/M on disk
TYPE	اكتب	Dump a file
USER	قسم	Change user area
-	انج	Switch to English commands

TABLE II
SAMPLE OF ARABIZED
MICROSOFT BASIC (KHAWARIZMI) STATEMENTS

BASIC	KHAWARIZMI
READ	اقرا
DATA	بيانات
WRITE	دون
OPEN	افتح
CLOSE	اغلق
USE	احضر
PUT	ابعث
FIELD AS	احجز كا
PEEK	محتوى
POKE	خزنحث
LOAD	حمل
SAVE	احفظ
IF THEN ELSE	اذا اذن والا
FOR TO NEXT	من الى التالي
GOSUB	اذهبرج
RETURN	عد
PRINT	اطبع
INPUT	ادخل
RUN	نفذ
SYSTEM	سلام

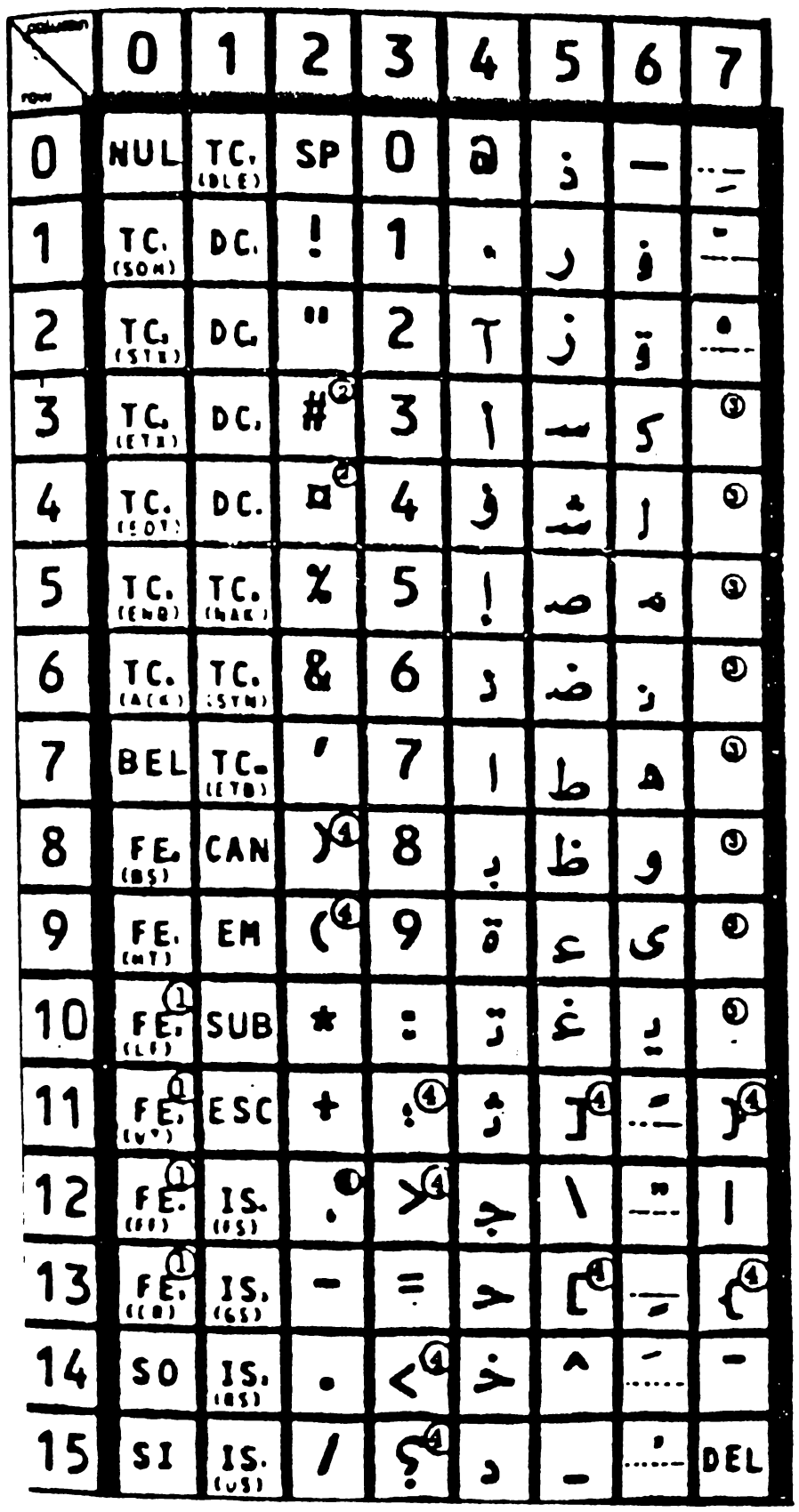


Figure 1. ASMO-449 Character Code.

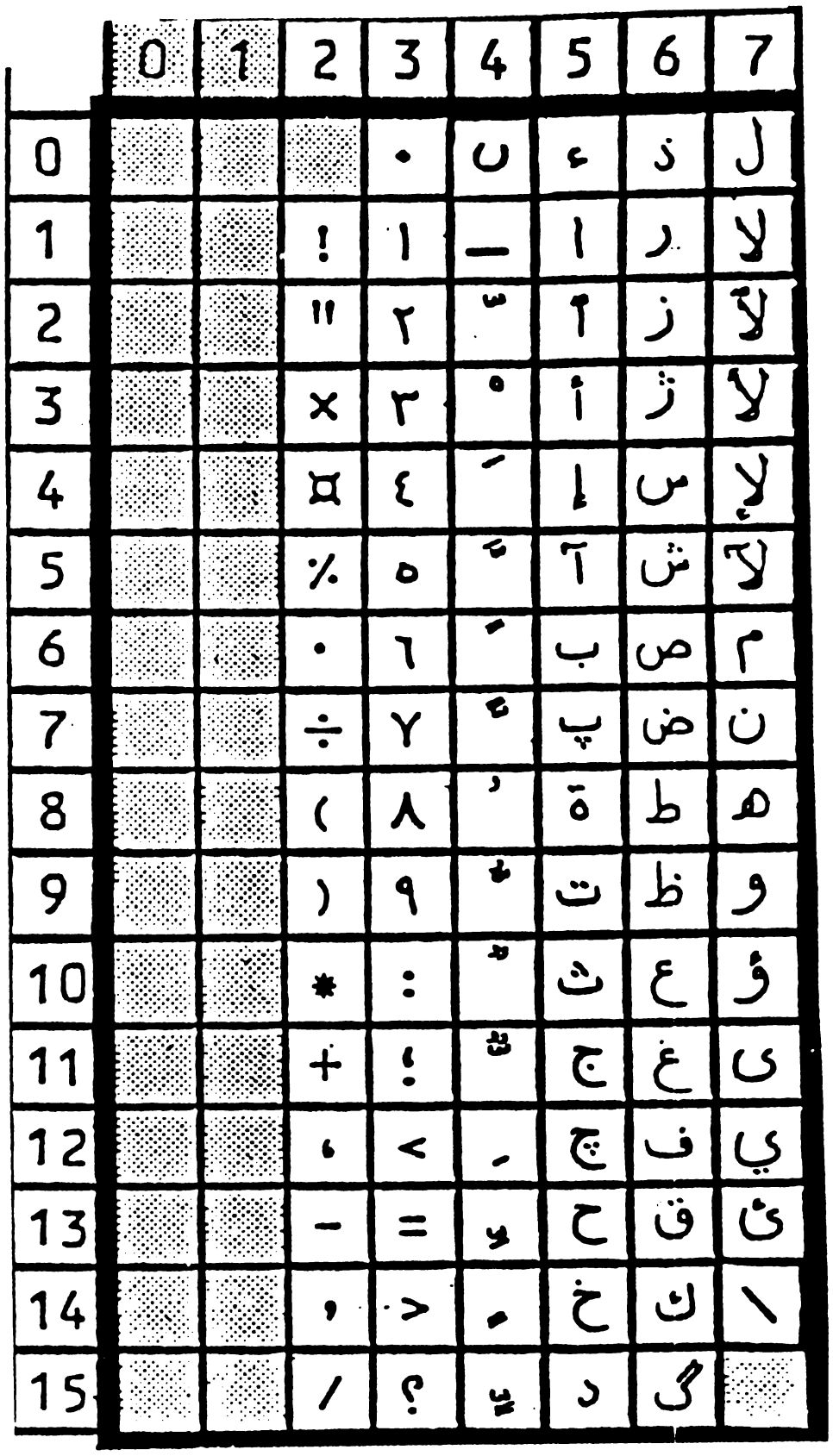


Figure 2. ARCII Character Code.

00 NUL 0	01 SOH 1	02 STX 2	03 ETX 3	04 EOT 4	05 ENQ 5	06 ACK 6	07 BEL 7	08 BS 8	09 HT 9	0A LF 10	0B VT 11	0C FF 12	0D CR 13	0E SO 14	0F SI 15
10 DLE 16	11 DC1 17	12 DC2 18	13 DC3 19	14 DC4 20	15 NAK 21	16 SYN 22	17 ETB 23	18 CAN 24	19 EM 25	1A SUB 26	1B ESC 27	1C FC 28	1D GS 29	1E RS 30	1F US 31
20 SP 32	21 ! 33	22 " 34	23 # 35	24 $ 36	25 % 37	26 & 38	27 ' 39	28 (40	29) 41	2A * 42	2B + 43	2C , 44	2D − 45	2E . 46	2F / 47
30 0 48	31 1 49	32 2 50	33 3 51	34 4 52	35 5 53	36 6 54	37 7 55	38 8 56	39 9 57	3A : 58	3B ; 59	3C < 60	3D = 61	3E > 62	3F ? 63
40 @ 64	41 A 65	42 B 66	43 C 67	44 D 68	45 E 69	46 F 70	47 G 71	48 H 72	49 I 73	4A J 74	4B K 75	4C L 76	4D M 77	4E N 78	4F O 79
50 P 80	51 Q 81	52 R 82	53 S 83	54 T 84	55 U 85	56 V 86	57 W 87	58 X 88	59 Y 89	5A Z 90	5B [91	5C \ 92	5D] 93	5E ^ 94	5F — 95
60 ` 96	61 a 97	62 b 98	63 c 99	64 d 100	65 e 101	66 f 102	67 g 103	68 h 104	69 i 105	6A j 106	6B k 107	6C l 108	6D m 109	6E n 110	6F o 111
70 p 112	71 q 113	72 r 114	73 s 115	74 t 116	75 u 117	76 v 118	77 w 119	78 x 120	79 y 121	7A z 122	7B { 123	7C ¦ 124	7D } 125	7E ~ 126	7F DEL 127
80 128	81 129	82 130	83 131	84 132	85 133	86 134	87 135	88 136	89 137	8A 138	8B 139	8C 140	8D 141	8E 142	8F 143
90 144	91 145	92 146	93 147	94 148	95 ~ 149	96 } 150	97 ¦ 151	98 { 152	99 ` 153	9A − 154	9B ^ 155	9C] 156	9D \ 157	9E [158	9F @ 159
A0 فراغ 160	A1 ! 161	A2 " 162	A3 # 163	A4 ◊ 164	A5 % 165	A6 & 166	A7 ' 167	A8 (168	A9) 169	AA * 170	AB + 171	AC , 172	AD − 173	AE . 174	AF / 175
B0 ٠ 176	B1 ١ 177	B2 ٢ 178	B3 ٣ 179	B4 ٤ 180	B5 ٥ 181	B6 ٦ 182	B7 ٧ 183	B8 ٨ 184	B9 ٩ 185	BA : 186	BB ؛ 187	BC < 188	BD = 189	BE > 190	BF ؟ 191
C0 ‟ 192	C1 ” 193	C2 آ 194	C3 أ 195	C4 ـَـ 196	C5 ـًـ 197	C6 ـُـ 198	C7 ـِـ 199	C8 ـْـ 200	C9 ـَ 201	CA ـُ 202	CB ـِ 203	CC ـُـ 204	CD ـًـ 205	CE ـٍـ 206	CF ـْـ 207
D0 ـَ 208	D1 ـُ 209	D2 ـٰ 210	D3 ـّ 211	D4 و 212	D5 ـٰو 213	D6 ص 214	D7 ~ 215	D8 ٥ 216	D9 ٩ 217	DA ١ 218	DB ـِ 219	DC ـّـ 220	DD ع 221	DE ـَ 222	DF — 223
E0 ء 224	E1 ى 225	E2 ا 226	E3 ب 227	E4 ت 228	E5 ة 229	E6 ث 230	E7 ج 231	E8 ح 232	E9 خ 233	EA د 234	EB ذ 235	EC ر 236	ED ز 237	EE س 238	EF ش 239
F0 ص 240	F1 ض 241	F2 ط 242	F3 ظ 243	F4 ع 244	F5 غ 245	F6 ف 246	F7 ق 247	F8 ك 248	F9 ل 249	FA م 250	FB ن 251	FC ه 252	FD و 253	FE ي 254	FF 255

Figure 3. RCTC Character Code.

is essential to interface with existing software written in English programming languages. There are thousands of excellent programs available in English programming languages for virtually all applications. If the source code and the permission to translate are obtained, these many programs can be easily converted for the Arab users convenience. Arabic Software & Computers, Inc. (ASCI) has a language converter from BASIC to Khawarizmi (and vice versa). Following is a brief description of this converter.

ASCI'S BASIC/Khawarizmi Converter:

This software package converts source programs written in Microsoft's MBASIC into RCTC's Khawarizmi. It is composed of two modules, TRAN1.COM and TRAN2.COM. Before a BASIC program can be converted to Khawarizmi, it is necessary that the program be free of bugs. The conversion is a three-step process:

1. Run TRAN1.COM. This program will take the BASIC source program (Let us call it TEST.BAS for illustration) and produce four files: TEST.KHB (contains converted Khawarizmi statements), TEST.ELT (contains a table of all literals found in the BASIC program), TEST.ESM (contains a table of all symbols, i.e., variable and file names) and TEST.ERM (contains a table of all comments).

2. Assuming that the converted program will be called CONVERT.KHB, translate the literals of TEST.ELT and place them in a file called CONVERT.ALT, translate the symbols of TEST.ESM and place them in a file called CONVERT.ASM and finally translate the comments of TEST.ERM and place them in a file called CONVERT.ARM.

3. Run TRAN2.COM. This program will combine TEST.KHB, CONVERT.ALT, CONVERT.ASM and CONVERT.ARM into the final Khawarizmi file called CONVERT.KHB ready to be interpreted by the Khawarizmi interpreter (after renaming it to Arabic).

(B) Program Translator Modification:

In this approach, the program translator is adapted to Arabic in addition to arabization of the programming language. In the previous example of BASIC, this means that the BASIC language and the BASIC interpreter are arabized. This approach requires attention to the details of the program translator. Among these details are:

1. The status of the higher-order bit for 8-bit-code systems. Many programs turn this bit on and off for various reasons. In systems which use this bit to distinguish between Arabic and English input/output, this action needs to be corrected.

2. Arabic file and variable names. Some translators allow a specific set of characters as valid characters in naming files and variables. The arabized program translator should extend this set to include the Arabic characters.

3. Full-screen editing. Care should be taken in arabizing segments of program translators which deal with cursor movements and full-screen operations. This is mainly because the diacritics, "hamza" and "lamm-alef" may occupy a number of display cells that are not equal to the number of characters they represent. For example, a "lamm-alef" with an upper "hamza" may be considered by a system to represent three characters while it only occupies one display cell on the screen. Cursor positioning and addressing need to be adjusted to relate to characters instead of display cells.

Examples of program translators arabized this way are Khawarizmi, Najla, and Dawaween. Table II shows some of the statements of Microsoft BASIC and their Arabic correspondents in Khawarizmi. Table III shows some of dBASEII commands and their arabization in Dawaween. Figure (4) shows a sample Khawarizmi program, Figure (5) shows a sample Dawaween program, and Figure (6) shows a sample Najla program.

```
٥  بيانات ٥٢٠٧-١٠١٠١٦٠٥٢٠١٢-١٠٨١٠،
١٠  ملاحظة ع =عدد الاعداد المراد ترتيبها
٢٠  اقرا ع
٣٠  ملاحظة عرف المصفوفة "ت" ذات الاتساع ع
٤٠  بعد ت(ع)
٥٠  ملاحظة اقرا الاعداد وعينها للمصفوفة ت
٦٠  من ل=١ الى ع
٧٠      اقرا ت(ل)
٨٠  التالي ل
٩٠  دون "ترتيب الاعداد الاصلي هو "
١٠٠  اذهبرج ١٠٠٠
١١٠  دون "خطوات ترتيب الاعداد:"
١٢٠  من س=١ الى ع-١
١٢٠      من ص=س+١ الى ع
١٤٠          ملاحظة اذا ت(ص) اصغر من ت(س) فاستبدل مكانيهما
١٥٠          اذا ت(ص)<ت(س) اذن بدل ت(س)،ت(ص) : اذهبرج ١٠٠٠
١٦٠      التالي ص
١٧٠  التالي س
١٨٠  دون "الترتيب التصاعدي للاعداد هو"
١٩٠  اذهبرج ١٠٠٠
٢٠٠  انه
١٠٠٠  ملاحظة بريمج يدون محتوى المصفوفة ت
١٠١٠  من ن=١ الى ع
١٠٢٠      دون ت(ن) ؛
١٠٣٠  التالي ن
١٠٤٠  دون
١٠٥٠  عد
```

```
نفذ
ترتيب الأعداد الأصلي هو
٥٢ -١٠١  ١٦  ٥٢ -١٢  ٨١  ١
خطوات ترتيب الأعداد:
-١٠١  ٥٢  ١٦  ٥٢ -١٢  ٨١  ١
-١٠١  ١٦  ٥٢  ٥٢ -١٢  ٨١  ١
-١٠١  ١٢-  ٥٢  ٥٢  ١٦  ٨١  ١
-١٠١  ١٢-  ١٦  ٥٢  ٥٢  ٨١  ١
-١٠١  ١٢-  ١  ٥٢  ٥٢  ٨١  ١٦
-١٠١  ١٢-  ١  ١٦  ٥٢  ٨١  ٥٢
-١٠١  ١٢-  ١  ١٦  ٥٢  ٥٢  ٨١
الترتيب التصاعدي للأعداد هو
-١٠١  ١٢-  ١  ١٦  ٥٢  ٥٢  ٨١
مستعد
```

Figure 4. A Sample Khawarizmi Program.

```
جلد
اد طالما صواب
امسح
نص
هل تريد
١    إضافة سجلات جديدة
٢    محو أحد السجلات
٣    البحث عن سجل
٤    تغيير بيانات أحد السجلات
٥    النظر في السجلات
٦    إصدار تقارير ؟
(للخروج اضغط على زر <اخرج> )

اخرالنص
ادخل "اختيارك هو" الى اختيار
اد حالة
    حالة اختيار = ١
        اد اضافة
    حالة اختيار = ٢
        اد محو
    حالة اختيار = ٣
        اد بحث
    حالة اختيار = ٤
        اد تغيير
    حالة اختيار = ٥
        اد نظر
    حالة اختيار = ٦
        اد تقارير
    والا
        ؟ "الرجاء إدخال رقم من ١-٦"
اخرحالة
اخراد
عد
```

Figure 5. A Sample Dawaween Program.

```
١٠   اطبع"                    شركة صناعات الكمبيوتر السعودية"
٢٠   اطبع"                    برنامج الهبوط على سطح القمر"
٢٥   اطبع"                                                  " : اطبع
٥٠   اطبع"السلام عليكم ... من مركز التحكم بالارض الى مركبة القمر"
٥٥   اطبع"اسكم على وشك الهبوط على سطح القمر..صير ان لديكم خللا فنيا"
٥٦   اطبع"في جهاز الهبوط الالي...وعليكم بالهبوط بالتحكم اليدوي"
٥٧   اطبع"لديكم ١٦٥٠٠ رطل من الوقود وستطيعون التحكم في الدفعات المعاد"
٥٨   اطبع"بمعدل سرعة استخدام الوقود (ك) كل ١٠ ثوان بحيث تكون (ك) صفرا او"
٥٩   اطبع"عددا بين ٨ و ٢٠٠ رطل في الثانية" : اطبع : اطبع
١١٢      س٠=٠
١٢٠  اطبع"الوقت اللازم للهبوط نحو الحاذبية هو = ١١٢.٥٥٢ ثوان وسرعة"
١٢١  اطبع"٤٠٠٨.٧٩ ميل في الساعة" : اطبع : اطبع
١٢٤  اطبع"لاحظوا الرادار وستمسى لكم حظا طيبا" : اطبع : اطبع
١٣٠  اطبع"بدء عملية الهبوط"
١٤٠  اطبع" الوقت";مروة(١٥);"الارتفاع";مروة(٣٠);"السرعة";مروة(٤٠);"الوقود";
١٤١  اطبع مروة(٥٠);"سرعة استخدام"
١٤٢  اطبع"بالثانية";مروة(١٥);"ميل،قدم";مروة(٣٠);"ميل/ساعة";مروة(٤٠);"رطل";
١٤٣  اطبع مروة(٥٠);"الوقود"
١٤٤  اطبع"    ";مروة(١٥);"    ";مروة(٣٠);"    ";مروة(٤٠);"    ";
١٤٥  اطبع مروة(٥٠);"
١٥٠  ل=٠ : ا=١٢٠ : ف=١ : م=٣٢٠٠٠ : ن=١٦٥٠٠
١٥١  ح=١-٠٣ : ز=١.٨
٢١٠  اطبع ل;مروة(١٥);جرص(١);" . ";جرص(٥٢٨٠*(١-جرص(١)));مروة(٣٠);
٢٢٠  اطبع جرص(٣٦٠٠*ف);مروة(٤٠);م-ن;مروة(٥٠);
٢٢١  اطبع"ك=";ك : اذا كان س=٠ متذلك ادخل ك : اطبع : اقصد ٢٧٠
٢٢٣  اقرأ ك : اطبع ك : اقصد ٢٠٥
٢٢٥  بيانات ٢٠٠،٢٠٠،٢٠٠،٢٠٠،٢٠٠،٢٠٠،٢٠٠،٢٠٠،٢٠٠،٨٠٠،١٥٠٨
٢٢٦  بيانات ١١،٠٣٧٢،١٠٠،١٠٠،١
٢٧٠  اذا كان ك<٠ متذلك ٢٧٢٠
٢٧١  اذا كان ك>٢٠٠٠ متذلك ٢٧٢٠
٢٧٢  اذا كان ك>٨ متذلك ٢٠٥
٢٧٣  اذا كان ك<>٠ متذلك ٢٧٢٠
٣٠٥      ت=١٠
٣١٠  اذا كان(م-ن)<١ا-٠٣ متذلك ٤١٠
٣١٥  اذا كان ت<١ا-٠٣ متذلك ٢١٠
٣٢٠      س=ت
٣٤٠  اذا كان(ن+س*ك)<=م متذلك ٣٥٠
٣٤٥      س=(م-ن)/ك
٣٥٠  بريمج ٩١٠
٣٥١  اذا كان ع<=٠ متذلك ٧١٠
٣٥٢  اذا كان ف<=٠ متذلك ٣٨٠
٣٦٠  اذا كان ع<٠ متذلك ٨١٠
٣٨٠  بريمج ٦١٠
٣٨١      اقصد ٢١٠
٤١٠  اطبع"نهاية الوقود في ";ا;" ثانية"
٤٤٠      س=(-ى+جذر(ى*ى-٢*ا*ج))/ج : ف=ف+ج*س : ل=ل+س
٥١٠  اطبع"الهبوط على القمر في ";ا;" ثانية" : و=٣٦٠٠ف
٥٢٠  اطبع"سرعة الاصطدام بالارض هي ";و;" ميل في الساعة"
٥٣٠  اطبع"الوقود الباقي هو ";م-ن;" رطل"
٥٣٥  اذا كان(-و+١٠)<=٠ متذلك ٥٥٠
٥٤٠  اطبع"الحمد لله على السلامة.. نهانينا" : اقصد ٥٩٠
٥٥٠  اذا كان(-و+١٠٠)<=٠ متذلك ٥٦٠
٥٥١  اطبع"الحمد لله على السلامة.. نرجو الانتباه مستقبلا" : اقصد ٥٩٠
٥٦٠  اذا كان(-و+٢٥٠)<=٠ متذلك ٥٧٠
٥٦١  اطبع"نهانينا على هبوط سيء" : اقصد ٥٩٠
٥٧٠  اذا كان(-و+٦٠٠)<=٠ متذلك ٥٨٠
٥٧١  اطبع"ربنا سلم ... لكن المركبة تحطمت" : اقصد ٥٩٠
٥٨٠  اطبع"لاحول ولاقوة الا بالله.. فلقد مات الجميع"
٥٨١  اطبع"في الواقع تم تكوين وادي جديد عمقه ";و*٢٧٧٧٧٧.١٠;" قدم"
```

Figure 6. A Sample Najla Program.

(C) Dubbing:

A third approach to arabization of program translators is dubbing. In this approach, the programming language remains in its original language, e.g., English, but strings, input and output are permitted in Arabic as well as the original language. This approach requires minimum adjustment to the program translator.

There are several arabized program translators in the market. We already mentioned Khawarizmi of RCTC, Dawaween of ASCI and Najla of Saudi Computer Industries. There is also a language called SAUDIA which is a dubbed BASIC language. A project for a Pascal-like language, called DAAD, has been launched(11). ASCI has started arabizing Turbo-Pascal into a language called "Bunyan". Descriptions of most of these languages are available(10,11). Saudi Computer Industries also carries dubbed versions of Pascal, C, Structured FORTRAN, COBOL, etc. and so does APTEC and Gulf Data, Inc. Table IV shows some of the statements of Turbo-Pascal and their arabization in "Bunyan". There are also very good programming tools available such as Arabdos 2.0 developed by Gulf Data, Inc., which facilitates bilingual programming, database management and is compatible with the ANSI 3.64-1979 command standards and BCON from ALIS which allows bilingual programming and is transparent to the application and the operating system.

IV. ARABIZATION OF SYSTEM UTILITIES

System utilities are programs that complement the operating system. They do tasks like copying files and disks, operating peripherals and checking disk storage, contents, and condition.

Another class of software programs that can be classified under system utilities is communication software. This type of software enables communication among similar or different computer systems and terminals. This makes it possible for the user to tap the enormous amount of information that can be stored on large systems, such as mainframe computers, using a small microcomputer as a terminal. The terminal will have the advantage of Arabic input and output. The terminal can also be used alone as an intelligent system. The programmer can also benefit tremendously from such software as he/she can use the enormous processing and storage capabilities of the host computer. ASCI has successfully developed an interface program, called "COM-7", which establishes communication between Alraed computers and other systems. It has been successfully used with Hewlett Packard, Digital Equipment Corporation and Data General computers and is expected to run successfully on all systems using RS-232 standard. IBM mini and mainframe systems can use "COM-7" by installing an RS-232 converter. Figure (7) shows the display of the main menu of options available in the COM-7 program. In addition, Gulf Data, Inc. has developed Arablink. It enables their Arabrite systems to emulate IBM terminals in both Arabic and English modes. Another 8-bit communication program is available from Saudi Computer Industries. It is called "Almirsal". It integrates telex operations with other functions such as word processing and electronic mail. It is bilingual and runs on Alpharabi computer systems. ALIS also announced that they have an 8-bit communication program.

Arabization of system utilities can follow normally after the arabization of operating systems and program translators (especially assemblers, since most utilities are written in assembly language).

V. ARABIZATION OF PRODUCTION TOOLS

Again, arabization of production tools such as word processors and database managememnt systems will follow easily once operating systems and program translators are arabized as described in the previous sections.

TABLE III
SAMPLE OF ARABIZED
dBASEII (DAWAWEEN) COMMANDS

dBASEII	DAWAWEEN
ACCEPT TO	انقل الى
APPEND BLANK	زد فارغ
CLEAR	جدد
COPY TO FIELD	انسخ الى حقل
COUNT FOR TO	احص لكل الى
CREATE	جهز
DELETE FOR	امح لكل
DISPLAY FILES ON	اظهر ملفات على
DO	اد
DO WHILE ENDDO	اد طالما اخراد
DO CASE OTHERWISE ENDCASE	اد حالة والا اخرحالة
ERASE	امسح
FIND	ابحث
INDEX ON	فهرس على
JOIN TO FIELDS	ضم الى حقول
LOCATE FOR	عين لكل
CONTINUE	استمر
PACK	كوم
QUIT	انه
RECALL	استدرك
REINDEX	اعدفهرس
SET ON/OFF	اجعل يعمل/لايعمل
SET TO	اجعل هو
SORT ON	رتب على
SKIP	اقفز
STORE TO	خزن في
USE	استخدم
WAIT	انتظر

TABLE IV
SAMPLE OF ARABIZED
TURBO-PASCAL (BUNYAN) STATEMENTS

TURBO-PASCAL	BUNYAN
READLN	اقراسطر
WRITELN	اكتبسطر
DELETE	احلف
CONCAT	صف
COPY	جزء
ASSIGN	عين
BLOCKREAD	اقراكتلة
BLOCKWRITE	اكتبكتلة
CHAIN	ادمج
EXECUTE	نفذ
RENAME	سم
RESET	هيئ
REWRITE	اعدكتابة
SEEK	ابحث
DELAY	اجل
GOTOXY	اذهبس
HIGHVIDEO	متوهج
LOWVIDEO	خافت
EXIT	انصرف
REPEAT UNTIL	كرر حتى
WHILE DO	طالما اد
FOR TO/DOWNTO	من الى/نزولاالى

```
برنامج الاتصال بين الحاسبات الآلية
جميع الحقوق محفوظة (١٩٨٥م)
مؤسسة البرمجة والحاسبات الإلكترونية العربية

                    قائمة الاختيارات
الاختيار                وظيفته
-----                   ----
<اخرج>ر          ارسال بيانات الى الحاسب البعيد
<اخرج>ف          فتح ملف ذاكرة لتلقي البيانات
<اخرج>ح          حفظ البيانات على القرص
<اخرج>ص          جعل صدى الإدخال يعمل/لايعمل
<اخرج>ط          تشغيل/إيقاف الالة الطابعة
<اخرج>ت          اعتبار/تجاهل الرموز الخاصة
<اخرج>ع          اعتماد الحاسب البعيد لبدء التشغيل
<اخرج>خ          انهاء البرنامج
<اخرج> <اخرج>    ارسال الرمز <اخرج> الى الحاسب البعيد
<اخرج>؟          رؤية هذه القائمة
```

Figure 7. Main Menu of ASCI's Communication Program.

In the Name of God, Most Gracious, Most Merciful بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

Praise be to God, the cherisher and sustainer of the worlds الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ

The Most Gracious, Most Merciful الرَّحْمَٰنِ الرَّحِيمِ

Master of the Day of Judgement مَالِكِ يَوْمِ الدِّينِ

You do we worship, and Your aid we seek إِيَّاكَ نَعْبُدُ وَإِيَّاكَ نَسْتَعِينُ

Show us the straight way اهْدِنَا الصِّرَاطَ الْمُسْتَقِيمَ

The way of those on whom You have bestowed Your Grace, those whose portion is not wrath, and who do not go astray. صِرَاطَ الَّذِينَ أَنْعَمْتَ عَلَيْهِمْ غَيْرِ الْمَغْضُوبِ عَلَيْهِمْ وَلَا الضَّالِّينَ.

Figure 8. A Document Produced by RCTC's Word Processor.

Word Processors:

A number of word processors which allow full-screen editing have been developed. Among them are:

1. Munasseq Alkalimatt. This bilingual word processor was developed by RCTC for Alraed computers. It uses special function keys to do the frequently needed tasks in text editing. This program is dependent on the operating system which is dependent on the hardware. A text formatter is available with this program to format the printed output.

2. Munazzem Alkalimatt. This is a product of ASCI. It is also bilingual and uses the same text formatter of RCTC. Instead of function keys, it uses control characters for full-screen editing.

3. Alkateb, from Saudi Computer Industries. This bilingual word processor works on Alpharabi computer series. It has many printing features such as boldfacing and underlining. It comes with a spelling checker, called "Almunjed", which can be customized to a particular vocabulary.

4. Arabstar-2, from APTEC. This is a full-feature, bilingual word processor that allows two-column editing, and "What-You-See-Is-What-You-Get" screen formatting. It even allows boldfacing and underlining on the screen. It is based on the famous WORDSTAR word processor and works on the arabized IBM PC computer.

5. Arabrite. In addition to all the good features of Arabstar-2, this bilingual word processor from Gulf Data, Inc., allows windows which permit editing of two files at the same time. There are customized versions of Arabrite for NCR PC, DEC Rainbow and other computers. There is also a program called "Multirite" which works with eight languages. Another program, called Arabedit, is specifically oriented to editing of source code programs.

6. Al-Tariq. This bilingual word processor was developed by M. Fakhouri and runs on an IBM PC computer with a graphics card and monitor. Arabization is done by installing a number of programs at the beginning of operation. It can display and edit the most complex character and diacritic combinations and print them using either a graphics or dot-matrix Epson-compatible printer.

7. Bilingual Editor (BED). BED is a line editor developed by Dr. Wilson Bishai, et al., and Education Development Center, Inc., and runs on the Apple IIe computer. It utilizes the graphic abilities of Apple and therefore requires no change to the original english-only hardware.

8. Tansiiq Alkalimaat. This is a full-screen Arabic word processor designed to run on the Apple Macintosh personal computer. It features "What-You-See-Is-What-You-Get" screen formatting, different font styles and sizes, mouse-controlled cursor movement and pull-down menus. It has been developed by Arabic Software Associates, Inc.

These word processors do a good job in editing texts and formatting the output to the printer. Figure (8) shows an example output from Munasseq Alkalimatt using an NEC letter-quality printer.

Database Management Systems:

Database management systems (DBMS) are not alike. Some are libraries of procedures that can be called from an application program, e.g., IMAGE/3000 of Hewlett Packard, others are menu-driven systems that require little or no programming, and others are programming languages with their own syntax and commands, e.g., SAS of SAS Institute and dBASE of Ashton-Tate. They are effective tools to develop application packages since they enable the design of specific systems without involving the programmer in database management tasks.

Among the many database management systems available for microcomputers, dBASEII has been the foremost. The raw version of dBASEII was developed at the Jet Propulsion Laboratory for NASA. This DBMS has been chosen by ASCI for arabization. The arabized version, called "Dawaween", was completed early this year (1985). One of the many conveniences of dBASEII is that it is a command language. This allows for custom programming and makes it easy to incorporate the query, report writer, form generator and editor parts of the DBMS in the language instead of having a separate utility for each of these important tasks. The next section explains in some detail the features of Dawaween.

A current project at the University of Petrolium and Minerals, Saudi Arabia(12) is undertaking the task of developing a language for processing bilingual databases called SRDL (short for Structured Relational Database Language). The project team plans to present two versions of the language, one version for Arabic and another version for English. Other bilingual database management systems available are "Alquaida" from Saudi Computer Industries, "Arabdbase" from APTEC and "Arabase" from Gulf Data, Inc. Arabase can interface with BASIC and COBOL compilers and allows time sharing.

Spreadsheets:

A computerized spreadsheet is a program that allows the manipulation of rows and columns of numbers and automatically adjusts the whole table as one or more of its cells are changed. This manipulation is performed by user specified arithmetic and/or mathematical functions and formula relationships. Spreadsheets are very useful in accounting, budgeting, finance, planning, and business modeling. ASCI has an Arabic spreadsheet called "Almunasseq Alhisabi". Figure (9) shows a display of a spreadsheet before and after an arithmetic operation. Another spreadsheet program offered by Gulf Computer Systems is called Arabcalc, and it represents an arabized version of the VISICALC program.

VI. THE DAWAWEEN LANGUAGE

Dawaween is an arabized version of the well-known database management system dBASEII. This version allows both English and Arabic in input and output. The language commands, however, are all in Arabic.

Dawaween is an application programming language particularly suited for database applications. It allows interactive, as well as program-controlled processing of database tasks, such as adding, deleting and updating records. It processes data quickly and efficiently without the need for writing lengthy and complex programs.

Dawaween has several features that ease the programmer's job. Some of these features are:

1. It is a structured language; there are no labeled statements and program execution is sequential. This simplifies the code and its testing and debugging too. In fact, the program could be composed of several modules, each given a name, and each module can be called any number of times from any other module.

2. It has the famous constructs which save the programmer time and effort. These constructs are:
   * IF - ELSE - ENDIF.
   * DO WHILE - ENDDO.
   * DO CASE - CASE - OTHERWISE - ENDCASE.
   * DO <Command file name>. This construct executes the commands of the module named after 'DO' in order.
   * TEXT - ENDTEXT. This construct dumps the enclosed text on the screen quickly. This saves a lot of WRITE statements in other languages.

3. It is interactive which makes program development and testing easier. It also allows quick computations and operation without the need to write a small program

مثال ٢ : عد(١١١،٠١١١،٠٦١١)=اب٠٦.

ملاحظة: الأماكن الفارغة تساوي صفراً.

قبل إجراء عملية العد

	عمود اا	عمود اب	عمود ات	عمود اث	عمود اج
صف ١		١١٢	١١٢	١١٤	١١٥
صف ٢	٢١١	٢١٢	٢١٢	٢١٤	٢١٥
صف ٣		٢١٢	٢١٢	٢١٤	٢١٥
صف ٤		٤١٢	٤١٢	٤١٤	٤١٥
صف ٥	٥١١	٥١٢	٥١٢	٥١٤	٥١٥
صف ٦		٦١٢	٦١٢	٦١٤	٦١٥

بعد إجراء عملية العد

	عمود اا	عمود اب	عمود ات	عمود اث	عمود اج
صف ١		١١٢	١١٢	١١٤	١١٥
صف ٢	٢١١	٢١٢	٢١٢	٢١٤	٢١٥
صف ٣		٢١٢	٢١٢	٢١٤	٢١٥
صف ٤		٤١٢	٤١٢	٤١٤	٤١٥
صف ٥	٥١١	٥١٢	٥١٢	٥١٤	٥١٥
صف ٦		٦	٦١٢	٦١٤	٦١٥

Figure 9. A Display from ASCI's Spreadsheet Program.

to do them.

4. It is a command language: Every statement is a plain Arabic command that is almost self-explanatory. By combining several of these commands, the programmer can form a module (or command file).

5. It allows the programmer to control system parameters, e.g., start and stop the printer. It also has several commands that do operating system tasks (like copying and erasing files and viewing the directory). Furthermore, Dawaween allows the programmer to examine and save the memory variables on a disk file.

Figure (5) gives an example program written in Dawaween.

VII. SUMMARY AND CONCLUSIONS

Software arabization requires the arabization of most software tools on top of which are operating systems and program translators. Only by doing so will the Arab user be able to program his/her computer entirely in Arabic without ever needing to know another language.

Few computer manufacturers have adopted this concept of arabization in developing their products and only for microcomputers. Research Computer Technology Corporation has a bilingual microcomputer, ALRAED, a bilingual word processor and an Arabic programming language, Khawarizmi. Arabic Software & Computers, Inc., has an Arabic database management system and language interpreter, Dawaween, a bilingual word processor and an Arabic communication program. Other firms such as Appropriate Technology (APTEC), Saudi Computer Industries, Gulf Data, Inc., Arabic-Latin Information Systems, Inc., Arabic Software Associates, Inc., and Gulf Computing Systems have a wide range of excellent, arabized software packages.

Standardization of Arabic informatics is needed to facilitate communication and compatibility between various computer systems.

Finally, in contribution to the enhancement of software arabization efforts, we suggest the following(13):

i. Emphasis should be placed on arabizing operating systems and program translators (programming languages).

ii. Existing, mature operating systems should be extended to process Arabic input/output and commands without disturbing the original, internal commands with a provision to interface the non-Arabic commands.

iii. Popular and potential programming languages should be arabized with the appropriate modifications done to the program translators. There should be a standard for syntax arabization.

iv. Every effort should be made to encourage Arabic software development.

v. Publications, conferences, seminars and literature translation should be invited and promoted.

vi. Coordination between various vendors and researchers should be achieved in order to eliminate duplication and redundancies.

vii. Interfacing tools to non-Arabic software should be encouraged and communication of Arabic needs to non-Arabic vendors should be emphasized.

REFERENCES

Note: References 1 and 4-13 are workshop papers presented at the Computer Processing of the Arabic Language Conference held April 14-16, 1985 in Kuwait.

(1) Mandurah, Mohammad M., "Architecture of an Arabic Microcomputer."

(2) ASMO-449, "Data Processing - 7-Bit Coded Arabic Character Set for Information Interchange," Arab Organization for Standardization and Metrology, Amman, Jordan, Oct. 1983.

(3) ALIS - RD404.2, "Arabic Reduced Code for Information Interchange," Arabic-Latin Information Systems Inc., Montreal, Canada, Oct. 1982.

(4) Abu-El-Haija, Ahmad I., "Codes and Standards: 7-Bit Coded Arabic Character Sets."

(5) Bouzid, M., and Arfaoui, A., "The Arabization of Informatics: The Tunisian Experience and its Contribution to Standardization."

(6) Khayat, M.G., "Printing Arabic Text Using Dot Matrix Printers."

(7) Shalie, Rastgow A., "Vowelized Text on an Arabic Terminal."

(8) Kaldarim, Osman, "Architecture of Arabic Computer."

(9) Mahjoub, Ahmed, "Arabic System Software Requirements."

(10) Hosni, Yasser, "Programming Languages for Arabic Applications."

(11) Khayat, M.G., "The Arabic Programming Language (Daad)." (in Arabic).

(12) Al-Suwaiayel, M. and Koyman, Kemal, "A Language for Processing Bi-Lingual Databases."

(13) Arafeh, S.A., "The Software Role in the Arabic Computer."

Arabic on Networks and Mail Systems

PIERRE A. MACKAY
Department of Near Eastern Languages
University of Washington
Seattle, Washington, USA

In the past decade, communication between computer systems has come to be almost as important as the capacities of the systems by themselves. Present day users of computing technology are no longer satisfied with the limitations of the large time-sharing main-frame on the one hand, nor with the autonomous, but restricted personal computer on the other. Companies which only a few years ago prided themselves on maintaining closed systems of proprietary software, now speak of "open systems" and "machine-independent software," as if these were the very heart of modern computing which, of course, they are. Computer systems communicate with each other not only locally, but across the world, through electronic mail programs, through file transfer, and through tightly coupled resource sharing.

As the Arabic-speaking world develops toward a wider and more general use of computers, it is natural to wonder what effect the use of the Arabic language will have on communications software. How much will have to be adapted, or rewritten when Arabic-speaking users begin to make large-scale use of network communications? It is pleasant to note that here is only case for which the answer is, "little or none."

The reason for this is a simple principle that underlies all successful network and communications programming. It is assumed that the tasks of establishing and maintaining a communications link can be divided into coherent groups, or layers, and that the layers can be arranged symbolically in a top-to-bottom hierarchy, starting from the user interface at the top, and running down to the physical communications-line interface at the bottom. The principle behind effective layering states that all control sequences in any higher layer must be treated as data by any lower layer. In systems which adhere to this principle, the fine points of natural-language input and output are completely irrelevant at all but the highest level.

Consider a very simple application of this principle. Text files are normally divided into lines, which usually emulate printed lines on a page. There are at least three ways of indicating the end of a line of text; in the Unix system, a line is terminated by a single line-feed character; in the DOS system used by many personal computers, a line is terminated by a pair of CR/LF (carriage-return and line-feed) characters; finally, in some systems based on ISO string-descriptor formats, a line is preceded by a character count and has no terminator character at all. These distinctions are significant at the top layers of a communications system, closest to the actual user interface, but at the lower levels, lines of text are terminated in some way that no user need ever be concerned with. When a line-feed system communicates with a character-count system, packets of data are exchanged which include some indication of line-end, but the software which exchanges the packets is or ought to be unaware of the particular line-end terminator which is used to present data to the user.

Owing to this careful isolation, a line from a CR/LF system can be transferred to an LF system without error, because the levels of software associated with making up and transmitting the packets for exchange between systems are unconcerned with the difference between the systems. Initial communication between systems tends to be always begun in this manner, since it is assumed that high-level (user-interface) commands will be passed between

systems in ISO character sequences. It is particularly revealing to consider the normal protocol for the transmission of non-character (binary) data.

A binary file, by definition, may contain any conceivable bit-pattern, and the bit-pattern for line-end is likely to be among those included. In this case, since the bit-pattern does not represent line-end, there has to be some what of ensuring that neither end of the file transfer performs any alterations on any part of the data. Across a network of heterogeneous systems it is too risky to depend on any pair of systems automatically getting it right by looking at the characteristics of the file itself. Therefore, the requester of services must be able to send a message to both the near and the distant system warning each of them to shut off line-end conversion. A message of this sort could also be used between homogeneous systems to speed up communications by avoiding redundant operations (conversion of line-end at the sender system and reconversion at the receiving system).

It is not the purpose of this paper to discuss general network protocols, and certainly not to get into the quasi-religious debate over the comparative virtues of the North American TCP/IP model or the more elaborate ISO-OSI Open Systems Interface. So long as the principle of layering is carefully adhered to, these two, along with any lesser systems, are far from being hopelessly incompatible. There are undoubtedly efforts at internet harmonization going on right now.

Our concern is to consider what is needed for two Arabic language applications to communicate across one of these networks, and the answer is, nothing at all, insofar as the file or message content of the exchange is concerned. If two personal computer systems could agree on a standard sequence (like the one that shuts off line-end conversion for binary files) which would switch each system into an Arabic text mode, they could begin communicating today. The only difficulty lies in the name-servers and addressing schemes.

Computers which talk to one another must have a way of finding one another, which is done through a system of names. For the convenience of the user these names are, in the best communications environments, associated with some sort of memorable allusion. (Locally, we use the names of South Sea islands for one network and of mathematicians for another. A neighboring commercial establishment uses the names of small furry animals for its network.) Nation-wide, names may be built up from a generic system of classification (e. g., separate genera for educational, commercial and governmental sites), and a further specification by something like locality, with an even narrower specification by agency, and a penultimate specification by machine.

At some time this gets transformed into a cryptic and much more compressed system of identifiers. For example:

Computer-Science.SSRC.Damascus.Academic

would be a possible address using a North American model. That would, still using the North American model, be translated into

987.65.43.2

a form which a lower level in the communications system could use efficently to identify the machine called. I do not have any sense of how an Arabic Language addressing scheme would fit into the operations of any known name-server, but I suspect it would not fit well. For the foreseeable future, users who chose to communicate in Arabic could see the entire text of a message in Arabic, but would have to put up with Latin-letter text for all the header and routing information. Ultimately, an applications program at both sending and receiving end of a communications system might be designed to display even header information translated into Arabic. It is easy to imagine an editor program for reading and writing mail which could perform the necessary translations into Latin-letter equivalents before sending and would do the reverse translation before displaying a received message.

The important point is that any necessary work in Arabizing communications systems for computer networks should concentrate entirely on customizing the ultimate end-user application. By adhering to the principles of layering, and presenting to the lower layers of a communications system only sanitized data, developers of Arabic Software can make free use of all the effective systems that have already been developed, without any need to alter them. There is perhaps no field where adherence to workable de facto standards is more important than in networking among heterogeneous computer systems. As in law, if it really doesn't work, then fix it, but otherwise, *stare decisis* is a good rule.

Generation of High Quality Arabic Computer Output

SHERIF SAMI and OUSSAMA ALAMEDDINE
IBM, Kuwait Scientific Center (KSC)
P.O. Box 4175
Safat, Kuwait

1. INTRODUCTION

Arabic is a very old language. Its origin lies in the Aramic, initially used by the Nabatean people. As other scripts of the semitic group to which it belongs, the Arabic flows from right to left and letter shapes vary according to the position of the letter in the word.

During the past 1000 years, 6 Arabic scripts emerged and are still most widely used. These are:

* Tholoth
* Naskh
* Requah
* Dewani
* Koufi
* Farsi

The rules governing the letter shaping and connecting within each of the above scripts are very complex. High quality Arabic script was therefore performed only by dedicated calligraphers.

When the print process appeared, Arabic typography had to settle on some machine compatible fonts. A long lasting process of simplification and standardization took place. This process has not yet ended, but many fonts based on a simplified version of the Naskh script are now commonly used by the Arabic printing industry.

The quality of Arabic computer typography is still far behind what is already achievable in Arabic machine prints.

The Kuwait Scientific Center (KSC) is investigating various techniques to produce Arabic computer outputs with quality comparable to that of the best text-book prints.

2. PARTICULARITIES OF ARABIC TYPOGRAPHY

In spite of the strong simplification of the modern Arabic text-book script, the Arabic particularities still complicate the process of its computerization. These are:

* Right to left writing
* The interconnection of adjacent letters (Cursiveness).
* different shapes for each letter according to its position in the word (alone, start, middle, end and more for logo fonts).
* Most font have highly curved letter shapes with the exception of koufi font.
* variable thickness of the letter shape drawing line.
* variable height to width ratio of letter shapes
* Characters integration (ligatures).
* Dots and diacritics
* Overlapping

Some of these particularities will be detailed in the following sections.

For Arabic quality output, most of these particularities are regarded as problems, because most of the techniques for the quality output generation are developed for Latin. The problems are raised while modifying an existing technique to handle Arabic or implementing a unique feature of the language.

At KSC, The generation of quality Arabic output has been addressed by investigating the following two problems:

* The generation of good quality Arabic fonts.
* The design of a formatter component that can handle Arabic particularities.

In the following, the two previous aspects will be discussed.

3. ARABIC FONT GENERATION TECHNIQUES

Computer typography makes use of various types of fonts for text output. Basically, these fonts might be classified into three groups:

1. Graphic font: The contours of each character are represented by a set of vectors.
2. Raster font: Each character is a rastered image of up to 600 pels/ inch.
3. Matrix bound font: Each character is represented by a limited set of N x M points.

In the following, the KSC approach to generate all three types of fonts is presented.

3.1 Generation Of Arabic Graphic Fonts

The first approach to generate graphic fonts used a digitizing tablet as input device. The chosen letter shape is properly laid on the tablet, and a manual process takes place to select the strategic points on the contours of each letter shape.

As an early experiment KSC applied the previous process on a complete Naskh font (corresponding to the version used by a local printing establishment), along with the associated diacritics, numbers and punctuations marks. Examples of the results obtained are shown in Figure 1 and Figure 2.

With regard to the tedious tablet digitization procedure, a new approach has recently been developed to speed up the font generation process and to improve the process of selecting the strategic points on the contours, improving the resulting quality. This approach depends on a digitizing camera and it might be summarized as follows:

* A group of letters (generally 16) is selected from a hand drawn character set. This group is entered as a rastered image through a digitizing camera (Figure 3 step 1).
* Using image processing techniques (detailed later), the contours of each letter are determined as closed polygons (Figure 3 step 2).
* These polygons are smoothed by recursive moving average to remove noise (Figure 3 step 3).

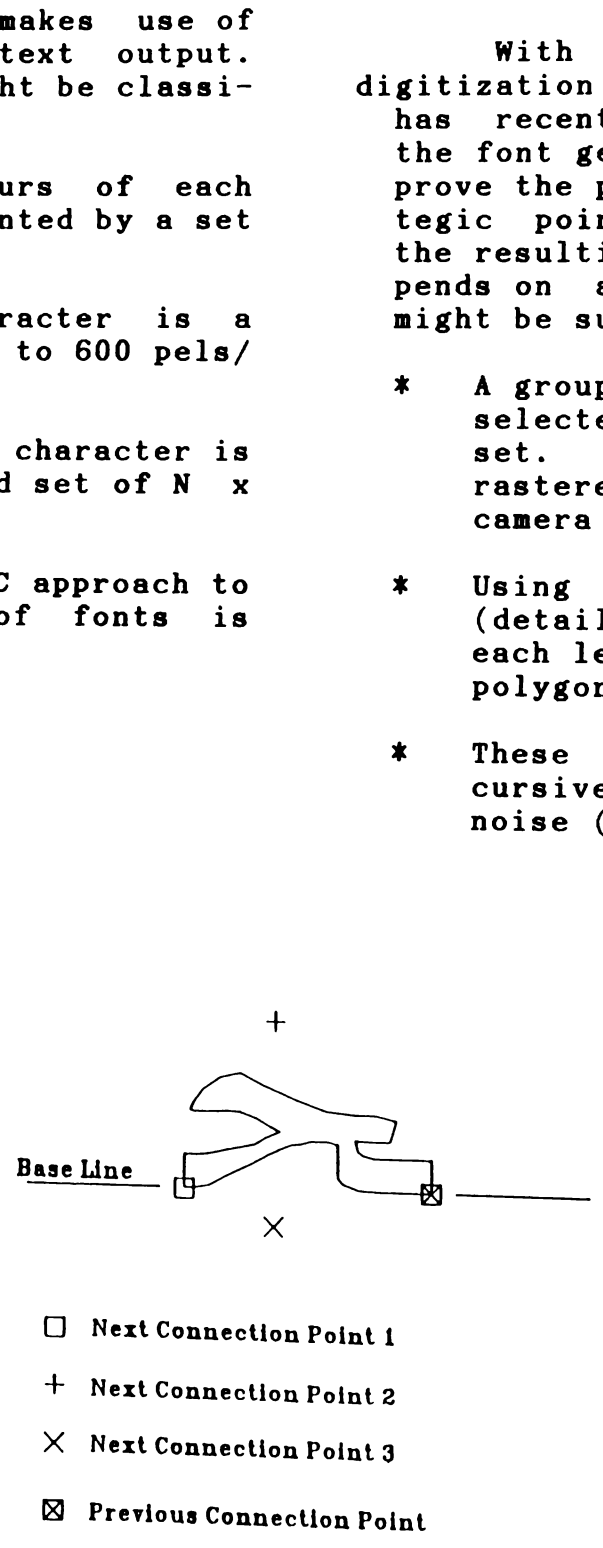


Figure 1. Components of the graphic model

* The polygons are vectorized with a predetermined error margin to minimize the number of vertices (Figure 3 step 4), (a large error has been selected purposely to show the single vectors in this figure).

* Finally, each letter is automatically calibrated to become compatible with other members of the font.

The ultimate result is a highly smooth and accurate font, which has been generated almost exclusively by automatical means.

This procedure was applied on an advertising Naskh version.

Figure 4 and Figure 7 show examples of the results obtained, demonstrating at the same time the letter shape smoothness even at strong magnification (compare with Figure 1 , where the font was produced by a digitizing tablet)

Figure 4 also shows how letters are interconnected.

Details regarding The process of determining the letter contours from its image are now given:

1. Thresholds: A raw image is a matrix (dimensions 426 by 512), which contains integers(indicating light intensity) between 0 and 255. However, in the case of letters, only two values are required. To get rid of all intermediate intensities, a number (threshold value) between 0 and 255 is chosen to separate black from white. Any point having a value below this number is called black (given the value 0) and any point above this number is white (given the value 1). This threshold value has been determined empirically to be around 80. (Actually it will depend on the lighting conditions when photographing the letters)

2. Contours: Now that the image contains only black and white points, the contours can be determined. This is done by shifting the image in all four directions (up,down,left,and right) and comparing it with the original. Where a black point meets a white point defines a point on the contour. Once these points are found their coordinates sequence is determined using the following algorithm:

a. Determine the highest and leftmost pixel P<0> and save it as first point in the sequence.

b. Define [P<0>] as a 3x3 matrix with center at P<0>

1) Select any of the contour pixels found inside [P<0>] (other than P<0> itself) to become P<1>. Then

* Save P<1> as the next pixel in the contour

* Erase all the points in [P<0>] except P<1>

Figure 2. Artistic use of graphic fonts.

* Repeat process with P<1> as P<0>, to get P<2>,etc.

2) If at step n (searching for P<n+1>) no contour pixel is found (It could be the last point in the contour) then

* Define [[P<n>]] as a 5x5 matrix with center at P<n>

* If [[P<n>]] is also empty then

- Define [[P<n-1>]]

- if [[P<n-1>]] is also empty then

-- P<n> is the last point in the contour

-- Append P<0> to the sequence

-- Go to step 1 to search for a new contour.

(Contours will be finished when no contour pixel can be found.)

- if [[P<n-1>]] is not empty

-- erase P<n>

-- delete P<n> from sequence

-- select any of the surrounding pixels as the new P<n>

-- erase all the points inside the augmented 5x5 matrix surrounding P<n-1>, (except P<n>)

-- continue with 3x3 matrices as in the start.

* if [[P<n>]] is not empty

- select any of the surrounding pixels to become P<n+1>

- erase all points in [[P<n>]] except P<n+1>

- continue with 3x3 matrices

3. Smoothing: The resulting contour will most probably contain small ripples as a result of the limit of the resolution of the camera and of the contour defining algorithm. To remove these ripples, a recursive smoothing approach is applied. In this approach, the coordinates of a point are replaced by the average of the coordinates of itself, the point before it, and the point after it. (this is always feasible, since the contour pixels are considered as a closed loop). By applying this method to all the points on a contour, a smoother curve is developed. The method is repeated on the new contour as many times as necessary to produce the desired smoothness.

4. Sharpening: The smoothing algorithm in the last section causes all sharp edges to be rounded. A method was developed to restore these sharp edges. (These sharp edges are especially important for the beauty of the Arabic characters). Each two consecutive points form a vector. If the angle between two consecutive

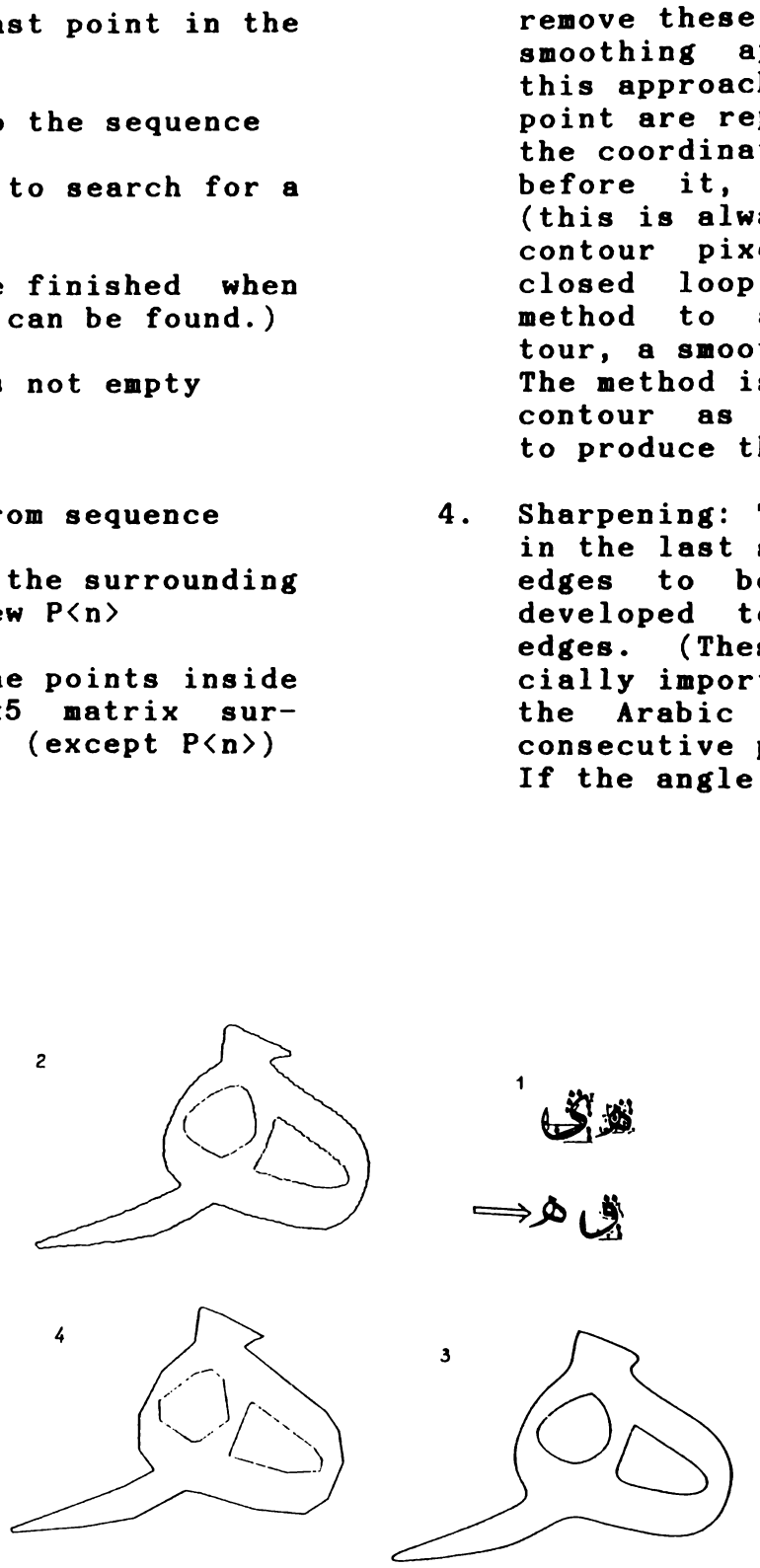


Figure 3. Phases of character digitization and vectorization

vectors is more than a given value (another magic number), then the second vector is marked for elimination. After repeating this process for every pair of vectors, those which are marked are removed, and are replaced with a point at the intersection of the vector before and the vector after the elimination. As a result, the round corner will be replaced by a sharp one.

5. Reduction: The number of vectors representing the image can be reduced by removing points which lie almost collinearly between two other points. Suppose matrix M contains the coordinates of the contour pixels. The following algorithm will remove points which lie close to a predefined line, and are thus not necessary. An error measure N will be used to determine if a point is close enough to be eliminated.

a. take the first pixel P<1>

b. initialize

 * MC=P<1> (minimum contour)

 * RC=P<1> (real sub contour)

 * PP=P<1>, P<1> (pixel-pixel segment)

c. Remove P<1> from M

d. take next pixel P<i>

e. RC=RC,P<i>

f. replace second pixel of PP by P<i> (PP will always contain only two pixels: first and last of real sub contour)

g. interpolate between the two pixels in PP to find as many points as in RC.

h. calculate distances between the points of RC and corresponding interpolated points. Find the maximum distance.

i. if this distance is smaller than N then

 * remove P<i> from M

 * if M is not empty go back to step 4

 * if M is empty then

  - take last element of RC and concatenate it to MC

  - end

j. if error is greater than N, then

 * take PS, the last but one element of RC

 * catenate PS to MC

 * initialize PP with PS in both rows

 * initialize RC with PS

 * go to step D

As a result, MC will be the minimum contour. It is a subset of the original

Figure 4. "advertising Naskh" with and without filling: The characters have been here automatically generated as per Figure 3

contour M. The maximum distance between MC and M is less than N.

3.2 Generation Of Arabic Raster Font

Besides our effort in the graphic representation of Arabic script, we are experimenting with the creation of rastered Koufi images using a color image processing system.

First, a model for synthesizing Arabic artwork in general, using dynamically generated rastered Koufi characters, has been developed. Each Koufi character is described by a matrix defining the vertices of the associated wire (lineal) skeleton, together with the places of special esthetical effects (triangular tips, circles, squares, etc...).

The rastered characters are constructed dynamically at any size and boldness by assuming a pen with circular cross-section moving along the character skeleton.

Additional effects, like shadowing, 3D, fringes, opalescence were developed, to create colorful and esthetic panels.

Although it is difficult to give a black and white reproduction of a color screen, Figure 5 is intended to give an impression of the results achieved.

3.3 Generation Of Matrix Bound Fonts

Matrix bound fonts are used in character mode on screens and matrix printers. As mentioned previously, the character is represented as a matrix of N columns by M rows, where the N and M are device dependent. The range of N and M can vary from 8 by 8 to 30 by 70 for some devices and still the technology is trying to increase the matrix size for better quality.

Any character is represented by setting some matrix elements to ON and others to OFF. Conventionally, the process of setting the points is done manually. The designer -Typically a computer professional, not a calligrapher- would use an empty matrix drawn on paper, or sometimes displayed on a screen, as in font editors. The corresponding results are often of limited quality.

Two of the various Arabic particularities affect significantly this font generation process. First, the Arabic high curvatures of characters impose the use of bigger matrices than those for Latin characters. Second, the drastic changes in width and height of character shapes reinforce the need for proportionally spaced fonts

As a step to simplify and improve the generation of matrix fonts for Arabic, an automation approach was investigated at KSC. This approach aimed at using the best possible font as input and automatically produces the best possible representation of the character in

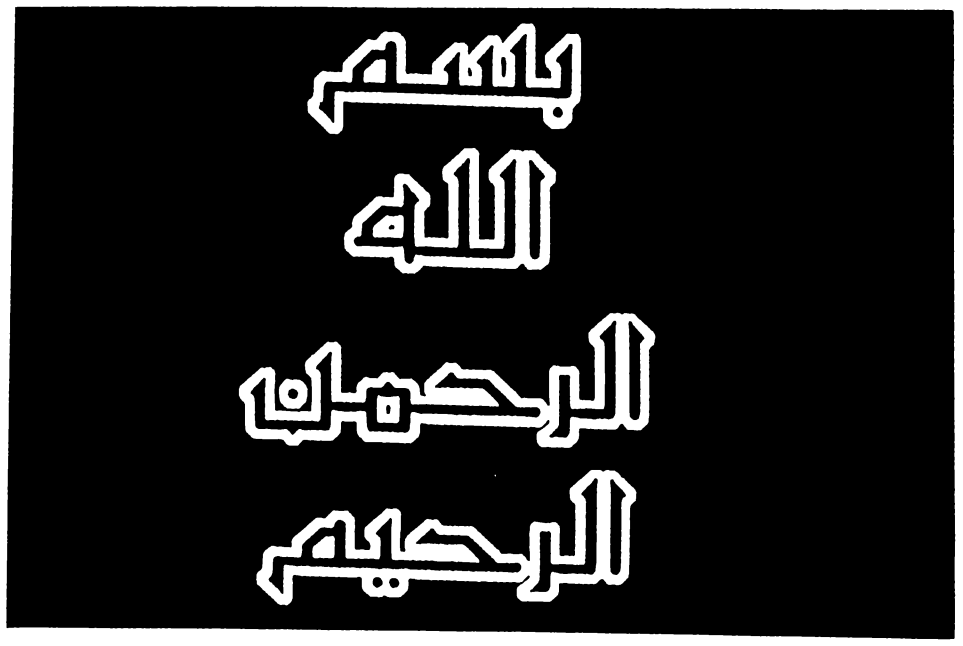


Figure 5. Koufi script with shadow effect: Black-and-white reproduction of color screen showing an example of the produced script.

the given matrix. In the following, details of the procedure are given:

* The first step is to digitize a good quality font. This process, including the transformation from a multi-level image to a binary image, has been detailed earlier under 3.1. The alignment problem is taken care of at the image capturing stage. Note that a sharp alignment is not essential. Once this binary font is created it could be used for the generation of several matrix fonts.

* The properties of the output matrix are given to the system in terms of the number of rows, number of columns, and the physical height width ratio. The height width ratio is only needed to preserve the proportionality of the output shape when the matrix element on the output device is not square. This data is used as a base to build an empty matrix for each character. This empty matrix is scaled to be overlayed on the image of the specific letter shape.

* This overlaying process takes into consideration the characteristics of the letter shape itself. For this process each Arabic letter shape is characterized by its position with respect to a base line, top line, bottom line, and left and right boundary lines. The characteristics are stored for each letter shape to automate the overlaying process (see Figure 6).

* As a result, every matrix element will contain a set of image pixels. On the output matrix the entire element will be set to ON or to OFF. In order to decide on this, a voting algorithm should be used. Several algorithms were investigated, and the most successful one was to check the center of the matrix element if it is On, then the vote will be ON for the matrix element, if not the vote will be OFF [4]. For a matrix element with even number of pixels in any direction the central two or four pixels are checked if any of them are ON.

* To produce proportionally spaced fonts, the same steps are followed producing first a non-proportional font. This font is then tailored by checking the proper width of each letter shape and removing the extra matrix columns.

A system based on the above procedure was designed and gave very good results for a matrix size of 9 by 12. For smaller matrices we would expect that some manual correction will be needed. For larger matrices the results were truly representing the master shapes as far as mapping is concerned.

4. ARABIC TEXT FORMATTING

A text formatter is a tool which allows the user to arrange some output text according to his requirements. Of course the formatting stage is always preceded by an editing one. In real time processing -as in word processors- the formatting is executed during the editing process. In post formatting, some commands are imbedded into the text during the editing process, which will direct the formatting process later.

Whether in real time or post processing, the formatting functions remain

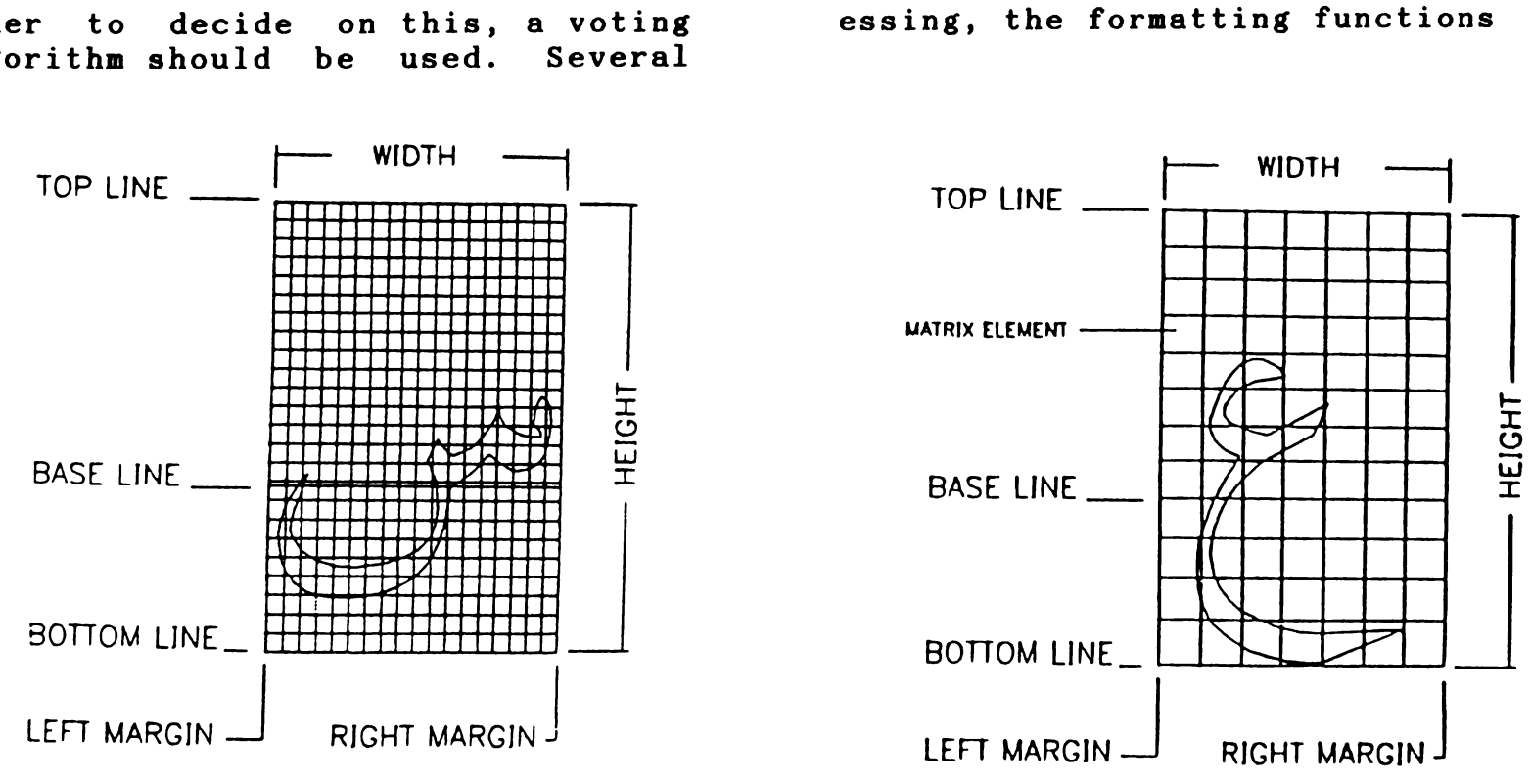


Figure 6. Positioning Arabic chracters within a matrix: Two examples of Arabic characters properly positioned in the generated matrix.

the same. We can group them into the following levels.

* Word level: for Latin, the formatter should check the best hyphenation position. For Arabic, the formatter should take care of the contextual analysis and the best elongation position in the word. No hyphenation is allowed in Arabic

* Line level: The line formatting is mainly responsible for the justification process using the given font specifications.

* Paragraph level: The formatting of a paragraph is responsible for distributing the text on various lines. This process is of course dependent on the given font and the line width for the given font, line width, and the left and right indentations.

* page level: In that part of the formatter the page width and height, foot note , header and trailer areas are controlled.

* Document level: This is where the structure of the document is controlled, such as titles, table of contents and indexes.

In the next section some of the formatting aspects will be discussed with respect to the Arabic particularities. Some of the KSC contributions in this area will be also given.

4.1. Arabic Drawing Model

An Arabic drawing model is an essential component to output quality Arabic text. It addresses the formatting on the character, the word and the line levels. The drawing model is applicable on both graphic or raster fonts generated by one of the previously discussed techniques. The model considers the interconnection of the successive characters and context dependent positioning of diacritical marks. This model is primarily valid for the Naskh and the Koufi scripts. The basic elements of this model have been worked out in an early KSC investigation and are reported in [1].

The various model components are shown in Figure 1.

This model has now been further improved in three respects by:

* Context dependant vertical and horizontal positioning of dots and diacritics, so that collisions with adjacent letters are avoided.

* Collision free overlapping of some letter combinations, like (Ra, Alef)

* Introducing of elongations which have various functions such as: Text justification, avoidance of collisions and embellishment (for the last purpose, a special algorithm for esthetical elongations was developed, more details regarding elongations will be given later, under "Arabic Line Justification").

Details regarding the first two points are now given:

Diacritics:

Characters which are written in vertically located positions (diacritics) are defined in the same way as ordinary characters, by a three column matrix defining the contours of the letter. In this matrix

* col 1 contains attributes. (0 for penup, 1 for pen down)

* col 2 contains x-coordinates of points on the contour

* col 3 contains y-coordinates of points on the contour

The location of the diacritic is defined by the character before it (the one upon which it is being placed) and the character after it. The position is defined in the following way:

1. Reset the diacritic by moving it so that its lowest point will be at the origin.

2. Move the diacritic up (down)

   * Detect the maximum (minimum) height of the preceding letter and add this value to the y-column of the diacritic

   * Also add (subtract) to (from) the y-column some buffer distance to avoid collision . (This buffer distance is defined as a global parameter and can be modified according to the user's desire)

3. Move the diacritic horizontally so that it will be centered above (below) the preceding letter.

   * Determine the maximum x-value of the diacritic

   * Subtract this from the horizontal space parameter of the preceding letter and divide the result by two. (This will give

the horizontal distance between the midpoint of the letter and diacritic)

* Add the result of the previous step to the x-values of the diacritic

4. Introduce elongation if following letter is too close

* Determine maximum (minimum) height of following letter

* Also determine the horizontal distance between the leftmost point of the diacritic and the rightmost point of the following letter.

If this height is higher (lower) than the minimum (maximum) height of the diacritic, and the horizontal distance is smaller than some fixed value, then introduce an elongation between the previous and the following letter.

Character Overlapping:

Letters which are not connected in a word sometimes need to be overlapped. This is especially true when the first letter is mostly below the line (like 'raa','zain' and 'wow') and the second is mostly above the baseline. In such a case, if no overlapping is made, excessive space would appear to split the word in two. The amount of overlapping needed depends on both letters and is defined as part of the font itself. Thus whenever a letter which cannot be connected from the left is encounterd, the letter after it is moved closer to it by a prescribed amount. If a diacritic lies between the two letters then the diacritic is also shifted (by around half the overlapping value) to avoid collision with the second letter.

If the second letter were a 'baa' or 'yaa', the shape of this letter is changed to make overlapping possible. Both these letters have dots below the base line, making it difficult to overlap them. By shifting their dots to the left and compensating this shift by lengthening their base (as if an elongation were added to it), a new shape which is easily overlapped is created.

4.2. Use Of Logo Fonts

Logo fonts are those fonts where more than one character can be merged together forming a distinct shape (ligature), Figure 8. This type of fonts which corresponds to traditional calligraphy is regarded by many to produce output of higher esthetic quality.

The early implementation of Arabic on computers did not use logo fonts because of the technological limitations and the nature of computer applications (payroll, stores, ..) which value speed over high quality. The modern computer technology managed to overcome many of the previous limitations with the evolution of all point addressable (APA) devices. Also the modern user applications developed to demand improved quality levels (text processing, publishing, ...).

لمْ يكُـــن الهَـــواء الـــذي

يُحيـــطُ بِنـــا فـــي يَـــومٍ مِـــنَ

الأيّـــامِ نَظيفـــاً ونقيـــاً . فلَقَـــد

كـــانَ هـــذا الهَـــواءُ دائمـــاً

ملوّثـــاً بِالمَـــوادِ الطَّبيعيـــةِ :

Figure 7. A formatted text using "advertising Naskh".

One of the important particularities of Arabic logo fonts is the way two or more letter shapes can be connected vertically causing a vertical shift with respect to the base line for all the preceding characters of the word.

The implementation of Arabic logo fonts in the printing industry is done by creating a separate block (slug) for each logo shape -combined letter shapes-, which results in a large number of blocks. In a computerized implementation for logo fonts KSC adopted a different approach by utilizing the computer power and logic to compose the logo shapes from their individual character shapes. In the following, details of this approach are presented.

* As a starting step a complete identification of all possible shapes of each Arabic character was done. This process depended upon the slugs of logo shapes used in the printing industry.

* The rules governing the use of each of these letter shapes were established by specifying the following properties for each letter shape:

  - The position where this letter shape can be used in the word. The four possible positions are alone, start, middle or end.

  - The previous and next letter or letters which will influence the selection of this letter shape.

  - The vertical shift level on which this letter shape can appear.

  - The amount of vertical shift caused by this letter shape affecting all the previous characters in the word.

  - the amount of horizontal shift for the letter shape, which will be less than a character width for logo shapes.

* An experimental set of logo character shapes was manually designed using an interactive graphic font editor in vector mode.

* An algorithm to perform the shape selection process was established together with the associated software package. The result is passed to a simplified drawing model, that displays the text in logo font using a graphic display interface.

The combined process of shape selection and shape composition, was found to be far more efficient if done on the word level starting from the last character, rather than from the first one (backward reshaping). The backward reshaping helped to simplify the process considerably for the following reasons:

* The last letter is always on the base line or what we call level one.

* As the reshaping proceeds, some letter shapes will cause level shift, that will be reflected immediately on the next character in the scan direction.

This approach can be implemented in real time processing on the word level (not on the character level). It is however useful in the post processing functions to produce final quality output on the printer or on the screen.

The simple reshaping of linear fonts (selecting one shape out of four: start, middle, end and alone) is no longer acceptable for a certain level of quality output. On the other hand, the full logo font should not be implemented for all computer applications, but only for those who need it. A more practical implementation is the use of a subset of the logo font to enhance the Arabic output quality. This subset is specifically those shapes that cause no

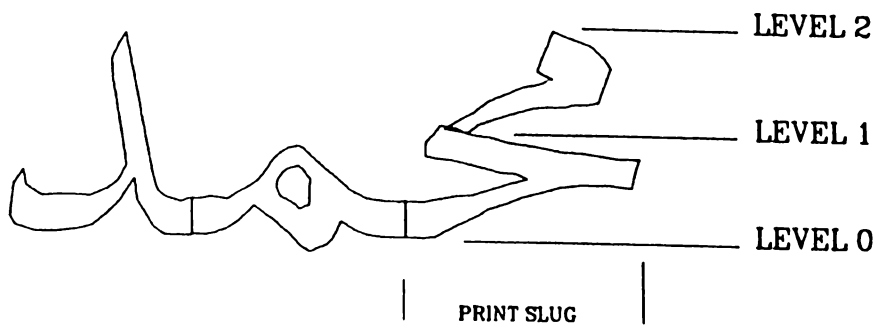


Figure 8. Arabic word in logo font: note the characters intersection and the level shift.

vertical shift for the preceding characters. This limited logo font use is implemented to different extents on some typewriters and in news paper printing.

4.3 Arabic Line Justification

Another particularity of Arabic is the justification of text lines, where the characters themselves are elongated instead of adding inter-words gaps, as in Latin. The selection of the best elongation position still depends greatly on the personal taste. Limited automation attempts were made to relief the user from the difficult task.

the following algorithm for the automation of Arabic text justification was developed and implemented at KSC.

* Set a priority level for each character shape which is allowed to be elongated. Our implementation used a 9 levels priority.

* Select whether the elongation position is to be before or after according to letter shape and user preference.

* Specify the maximum allowable elongation for each selected character shape.

* A further improvement is to add a control to prevent the elongation in some conditional cases (like the kaf followed by alef or lam followed by lam).

* The distribution of extra spaces in a line is done then by the following algorithm

  - Scan the line to detect all possible elongation positions in all the words of a line.

  - The priority of each possible elongation position is checked against the predetermined priority table.

  - All Possible elongation positions are sorted according to the assigned priority.

  - The extra spaces in a line are distributed to the highest priority elongation positions limited by the maximum elongation for each position.

  With all the previous information the distribution process can be controlled to give a higher preference within the same priority group to the left to right occurrences or vice versa. It can be also controlled to allow for multiple or single elongation in each word. The user can also control the amount of elongation and the priority by simply changing an entry in the control table.

4.4 Bilingual Text Formatting

When we address the pure Arabic document formatting, we will find that the formatting functions are quite similar to those of Latin documents with the exception of the text flow direction. Unfortunately, bilingual text formatting became a necessity for modern users. This requirement adds more burden on the functions of Arabic formatters, as they have to consider the flow of Latin text into an Arabic paragraph, and vice versa.

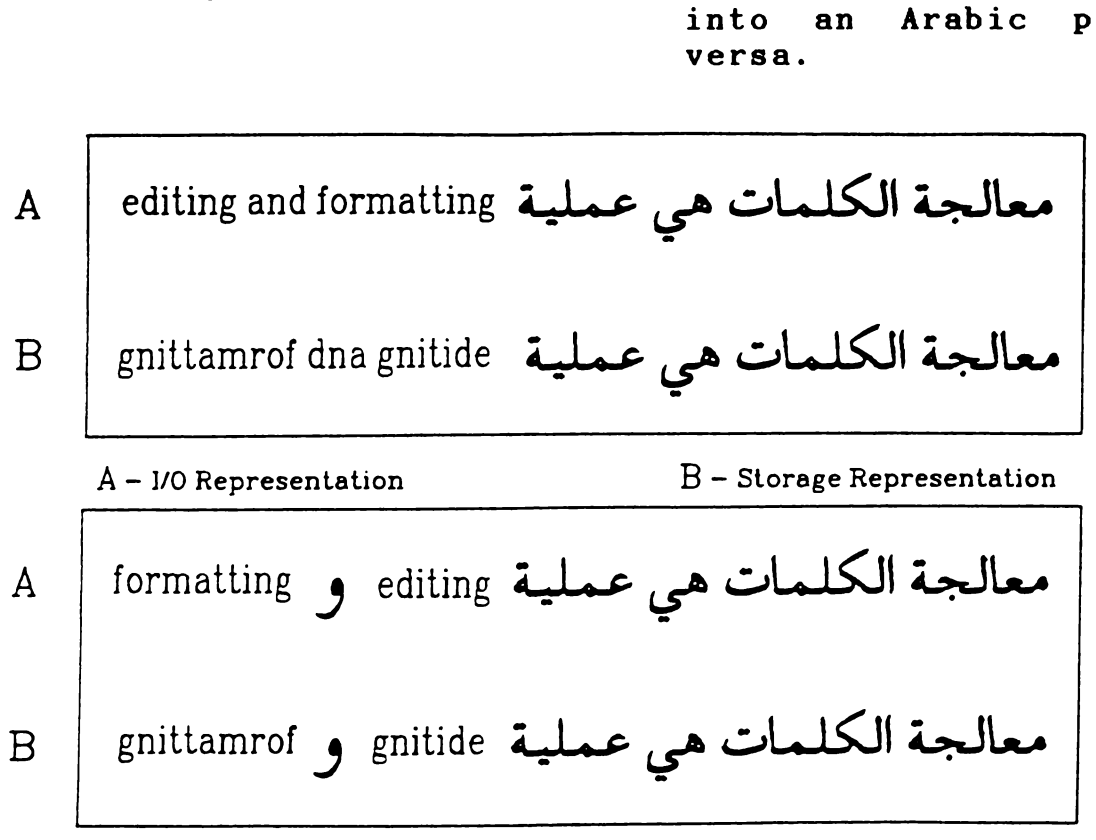


Figure 9. A) Is a I/O representation. B)Is a storage representation.

For bilingual text formatting where Latin text is imbedded into Arabic, the flow of text should obviously be controlled. Our approach [5] (to control the flow) depends mainly on the definition of two different representations of the text:

* The storage representation, where all the text (Latin and Arabic) flow in the same direction.

* The input/output representation, where every text is represented in it's proper direction, see Figure 9.

All the formatting functions will be performed on the storage version of text, which will be normal formatting without considering the bilingual text.

To change the text from the input/output representation to the storage representation, the following algorithm was investigated at KSC: (In this example we assume an Arabic text containing some Latin portions. The Latin text to be changed to flow in the same direction as Arabic).

* Identify the first occurrence of a Latin character and flag the start of Latin string.

* Scan the paragraph lines forward for the first occurrence of an Arabic character, or the end of the line.

* If Arabic character, scan backwards for the first Latin character and flag the end of Latin string. If it is the end of the line no need to scan backwards, just flag the end of Latin string.

* Reverse the identified string to flow in the same direction of Arabic

* Repeat the process for all Latin strings in the paragraph. This will produce the storage representation of the text.

* Perform paragraph formatting as if the text is all Arabic.

* For input/output representation repeat the same process of isolation and reverse on the formatted text.

This algorithm will take care of all the imbedded punctuation marks which are normally common to both languages. The same procedure can be implemented on a Latin text containing Arabic portions. The Arabic text should be reversed to flow in the direction of Latin text. A similar algorithm will identify and reverse Arabic portions instead of Latin ones.

The implementation of this approach can be either on the terminal level or at the CPU level when the beginning and end of text in a given language is not identified.

5. FUTURE PROSPECTS:

There are still many open areas with important fallouts. KSC is currently working on the following subjects:

* Semi automatical creation of Arabic wire fonts associated with existing bold fonts, using image processing techniques. Such a font is practical for quick drawing and for teaching how to write Arabic. It might become the starting basis for creating fanciful writing models, by assuming a certain pen shape and a certain hand movement.

* Migration of some or all of the above techniques to micro processors.

* The implementation of logo fonts on all point adressable screens.

REFERENCES

1. Ayman Al-Asadi, Jesus Rueda, Graphic Representation Of Arabic Alphabet, KSC report No. 008, 1984.

2. Donald Knuth, The Metafont Book, Computers And Typesetting, vol 1, Addison Wesley Co. 42000

3. Donald Knut, The Texbook, Addison Wesley Co ,1984

4. Charles Bigelow and Donald Day, Digital Typography, Scientfic American, pp.94-105, August 1983.

5. Joseph D.Becker, Multilingual Word Processing, Scientfic American, pp. 96-107, July 1984.

APPLIED ARABIC LINGUISTICS FOR INFORMATICS

Impact of Linguistics on Informatics

SAMER ATTASI
IBM, Kuwait Scientific Center (KSC)
P.O. Box 4175
Safat, Kuwait

1. INTRODUCTION

As computers are no longer confined to the circles of data processing specialists and scientists, and new applications are reaching people of various disciplines, various professional levels and various age categories, there is an obvious need for a more 'natural' environment for man machine communication. This requires the development of

1. Advanced applications such as text proofing, natural language query systems, computer translation, intelligent language education etc.

2. Appropriate I/O techniques including speech, optical and advanced text entry functions

3. Appropriate system environments (architectures, programming tools)

We shall briefly describe the basic linguistics tools as they are or can be implemented on computers, and then analyze their impact on the developemnt of advanced applications, I/O techniques and system environments.

2. LINGUISTIC COMPONENTS FOR THE PROCESSING OF NATURAL LANGUAGES

The basic linguistic components considered here are meant to give computers the ability to 'understand' a natural language sentence (entered or stored in the form of text) and/or generate such a sentence from a representation of its 'meaning'.

Early programs such as DOCTOR by Weizenbaum in 1965, and STUDENT by Bobrow in 1962 could sustain a 'natural conversation' where the appearance of understanding and sentence generation was derived from a clever use of keywords. Using pattern matching techniques in LISP [1] or through the unification algorithm technique [2] in PROLOG, a program like DOCTOR would be a simple exercise today.

However there is no shortcut to guenine analysis and appreciation of the 'meaning' of a natural language sentence. Following are the basic components required for that purpose.

2.1 Lexis

First a comprehensive lexical data base must be available to identify the words of the sentence and provide related linguistics information for further analysis of the sentence.

Efficient storage and retrieval of lexical information is highly language dependent. In the case of highly inflected languages morphology can be used to compact the lexical data to a minimum.

In the case of Arabic, a lexical data base consisting of the verb roots properly classified according to morphological generation rules, plus a limited number of closed functional sets, can be built. In conjunction with a morphological analyzer [3], this lexical data base becomes an operational and highly efficient tool for word analysis, and can also provide limited syntax information (grammatical values of words, gender and number of nouns).

Arabic also has the property that semantic transformations can often be associated with morphological transformations. This matter requires further research and analysis. Ideally it could lead to a classification of roots according to their semantic transformation classes, and would thus provide an integrated root based lexical ata base.

2.2 Syntax Parsing

The basic function of a parser is to map a sentence from a language onto a valid 'grammatical structure'.

In its simplest form, the parser consists of a limited set of combination rules whose arguments are explicitly the elements of the allowed vocabulary, and which are mapped by the combination rules onto a unique tree like structure. Such parsers are used in a number of traditional applications involving a 'command language'.

Natural language parsing [4] is much more complex and requires the development of new theories and new techniques for syntax parsing.

On one hand, sentence structures are versatile in natural language; neither the unicity nor the tree like structure assumptions are valid any more. Furthermore in provision of the further step of semantic analysis, transformational grammars can be used at syntax analysis level to transform the input sentence into a 'canonical form' structure, more amenable to semantic analysis

On the other hand, natural language vocabulary is too large to be explicitly used in grammar rules. This implies that grammar rules must be applied to grammatical values provided by the lexical data base. This implies an interdependence in the design of the lexis and of the syntax parser.

This interdependence may be even higher in the case of Arabic than in the case of other natural languages. First, because part of the syntax analysis is performed at the lexis level due to the phrase like structure of Arabic words. Second because the lack of voyels in standard Arabic text creates a large number of homonyms, which multiplies the number of ambiguities to be resolved by the parser.

2.3 Semantic Analysis

The semantic analyzer uses semantic attributes provided by the lexical data base, and applies semantic rules to map the sentence into a valid semantic structure. Obviously linguistics play a major role in the definition of semantic attributes and semantic structures [5] [6]

We have mentionned previously that the potential of using morphological transformations as a support for semantic transformations could lead to an efficient Arabic lexical data base. However, the same strong degree of interaction (as with syntax analysis) between the lexical data base and the semantic analyzer is expected to arise in Arabic for the same reasons (homonyms, phrase like words...).

2.4 Sentence Generation

Sentence generation differs from text understanding in 2 respects

* The language set covered by the application is under the control of the computer, and therefore some complications could be avoided.

* The computer has to generate the text of the sentence from a representation of the 'meaning'. The origin and nature of the representation vary widely with the application. In computer aided translation the 'meaning comes from the origin language sentence in the form of a semantic structure. In a data base query system, it comes from a semantic representation of the data base information.

3. IMPACT ON ADVANCED APPLICATIONS

Text understanding provides a new dimension to the scope and efficiency of advanced applications

3.1 Text Proofing

A good text proofing application should be able to highlight misspelled words, awkward syntactic constructions, improper contexts etc.. and provide error correction aids. To be of any use, the proofing should be reliable, in the sense that no errors are passed unnoticed and a minimum of false alarms are generated.

Consider the 'simple' task of word spellchecking in Arabic. First the application should have access to a comprehensive lexis to check against it every word in the proofed text. A comprehensive lexis with all the inflected forms in Arabic, including the attached pronouns, articles and prepositions would require several hundred thousand entries, if not millions. A lexis constructed as described in paragraph 2.1, can represent a storage compaction of more than 2 orders of magnitude. Besides

the storage saving, search time is dramatically reduced.

Further the lexis based on the morphological rules can detect 'micro-syntax' errors at the word level (before searching in the lexical data base). Morphological rules can also provide syntax information (for example gender and number of nouns) which can be used in syntax checking (verb/subject and noun/adjective accords etc...)

3.2 Natural Language Query Systems

Without solid linguistics components, a natural language query system can be built using pattern matching techniques. However such a system will not really analyze the 'meaning' of the sentences, and will have two major limitations

* The range of sentences it can 'understand' is limited and arbitrarily defined by the pattern matching technique.

* The system is not reliable and can totally misinterpret a question.

Naturally there are 'smart' programs which can dialogue, giving precise answers when very specific patterns occur, and very general answers otherwise. This approach is however of limited use in real life application

A lexis, a natural language parser and a semantic analyzer provide the proper frame to avoid the above shortcomings.

3.3 Computer Translation

Computer translation can be done without solid linguitics components, on limited sets of a natural language. This can be of use in the translation of simple command languages and message generation components, and could simplify for example the Arabization of the software interface of some existing applications.

However, this approach is obviously not appropriate for text translation, where the vocabulary is large and syntactic constructions are unconstrained.

A solid text translation component requires efficient and comprehensive lexicons, a parser based on a transformational grammar for the input language, a semantic analyzer, and a generator for the output language.

3.4 Computer Aided Instruction (CAI)

Using computers in education has emerged with the development of personal computers as a very important potential application area

CAI can deal with teaching or training on algorithmic thinking, or can represent a teaching aid in standard Highschool and University curricula (Mathematics, Physics, Language..) or can represent a professional self education facility (Data processing courses etc...).

In all the cases, language intelligence provides a great plus to the CAI application.

In language teaching for example a system knowing the vocabulary and the rules of grammar (lexis and syntax parser) is much more powerful than a system with predefined grammar questions and answers.

In other disciplines, language components can make the interaction with the student richer and more natural. The answers can be analyzed for what they really mean, without the restrictions of an artificial language. Free comments could be given by the student which could be analyzed and be taken into account in the teaching strategy.

4. IMPACT ON ADVANCED I/O'S

The purpose of advanced I/O's is to make the interface between computers and users more effective, more flexible and more natural.

It is noteworthy that advanced I/O is a domain of interaction of linguistics with other fields of artificial intelligence, such as for example pattern recognition. Thus pattern recognition problems arise in Speech and Optical recognition, while 'syntactic' techniques are in common use in pattern recognition problems.

4.1 Text Input

In a 'text understanding' computer environment, it can be expected that the text understanding components would be used to check the user's current input for spelling and also for ambiguities (at word, phrase or sentence level).

In the case of Arabic, where data is entered without short voyels and diacritical signs, the user could also be prompted in case of ambiguities, to select a diacritics setting among a number of alternatives.

4.2 Optical Document Entry

The transformation of a text in a scanned document into a string of character codes is a language dependent process, usually referred to as character recognition [7]

The basic process of character recognition is a pattern recognition process where all the techniques of that discipline (statistical, topological, syntactical) are applicable.

On the other hand, regardless of the pattern recognition technique used, proofing the recognized text is necessary. This requires a dictionary, syntax checking and other linguistics tools (error correction aids..).

4.3 Speech Recognition

The transformation of spoken voice information into the corresponding text information is known as speech recognition [8].

The involvment of computational linguistics in speech recognition systems increases with the language set to be 'recognized'.

While recognizing a handful of distinct words is merely a pattern recognition exercise, recognizing continuous speech with a vocabulary of a thousand words requires the use of advanced linguistics techniques whereby for example a phrase is 'recognized' first at the phonetic level, then checked at the word level using a dictionary, then at the syntax and semantic levels using appropriate rules. The process is automatically reiterated bottom up and top down until it hopefully converges to the correct interpretation.

4.4 Speech Synthesis

Speech synthesis is the transformation of text into spoken voice output [9].

Phonology is essential in the definition of the basic sounds (e.g allophones) and of the prosody rules. The prosody rules themselves are dependent on lexical, syntactic and semantic information.

5. SYSTEM ENVIRONMENTS

Trying to address computational linguistics topics and more generally artificial intelligence topics, using traditional system architectures and programming languages, has shown a number of limitations, which are being addressed today by the Computer Science research and development community.

5.1 Programming Languages

Traditional 'high level' programming languages were designed to cope with the requirements of Scientific computing or business applications. Even in those areas new requirements have emerged, but ways to handle them are now understood.

In Scientific computing, versions of Fortran and other languages with vector and matrix manipulation capabilities have been implemented including imbedded algorithms for improving and evaluating calculation accuracy.

In business applications the trend has been towards the development of general data base systems with data base management languages, saving a great deal of effort for applications development.

The manipulation of 'knowledge' and 'logic programming' required by linguistics and in general by artificial intelligence applications pose new requirements on programming languages, such as

* Extensive symbol manipulation operations for example the use of 'unification' techniques in pattern matching on words and phrases.

* Ease in the representation of trees and nets which are the usual structures to store syntax and semantic information

* Ease in the manipulation of nets for example the availability of backtracking techniques in the search processes, which allows to resume an unsuccessful search at an earlier node.

These requirements have lead to the definition and use of new programming

languages such as LISP and more recently PROLOG.

PROLOG, whose 'naked' structure reveals its logic programming nature (the user workspace contains the knowledge base where assertions express facts, axioms express relations and goals express inquiries), was initially intended [10] as a tool for processing natural language. It is becoming the basis for R&D in artificial intelligence as well as for research in new machine architectures.

5.2 System Architectures

The performance of machines used in traditional applications are evaluated in MIPS (Million machine instructions per second) and Megaflops (Million of floating point operations per second).

Using todays LISP and PROLOG interpreters running with traditional machine architectures, it takes approximatively 10 million instructions to perform a comprehensive analysis of an average English sentence. This means 1 second CPU time on a 10 MIPS machine.

Orders of magnitude improvements are needed to implement on a wide scale, advanced interactive text understanding systems whose preformance should rather be measured in KLIPS (Thousand Logical inferences per second). A great deal of improvement is expected in the following areas

* Improved techniques for knowledge representation.

* Improvements in the procedural control of the execution of PROLOG programs. This could be achieved through the development of optimizing compilers able to generate efficient executable code, minimizing for example the backtracking to the necessary minimum etc..

* Extraction of 'parallellism' in the execution of PROLOG programs, and definition and implementation of appropriate machine architectures.

6. CONCLUSION

We have attempted to describe briefly the impact of linguistics on informatics. In fact, it would be more correct to speak of 'interaction of linguistics with informatics' since the impact goes in both directions. The involvment in computational linguistics is raising questions on knowledge representation and its relationship with natural languages, which are sometimes of high philosophical content, and can be of interest to everyone.

REFERENCES

[1] Winston P. H. and Horn B. K. P., LISP ,Addison Wesley Publishing Company, 1981.

[2] Nilsson N. J., Principles of Artificial Intelligence ,Tioga publishing Company, Palo Alto, California, 1980

[3] Thalouth B. and Al Dannan A. A Comprehensive Arabic Morphological Analyser Generator , Proc. Arab School on Science and Technology, Zabadani Syria, 1985

[4] Jones K. S. and Wilks Y. Automatic Natural Language Parsing ,Ellis Horwood Limited, 1983.

[5] Hayes P., Semantic Markers and Selectional Restrictions, in Computational Semantics ,ed. Eugene Charniak and Yorik Wilks, pp. 41-55, North-Holland pub. comp., 1976

[6] Scragg G., Semantic Nets as Memory Models, in Computational Semantics ,ed. Eugene Charniak and Yorik Wilks, pp. 101-129, North-Holland pub. comp., 1976

[7] Ullman J. R., Advances in Character Recognition, in Applications of Pattern recognition ,ed. K.S.FU, pp. 197-236, CRC Press, Boca Raton, Florida, 1982.

[8] Lea A. and Wayne, Trends in Speech Recognition ,Englewood cliffs, N.J. Prentice Hall, 1980.

[9] Bristow G., Electronic Speech Synthesis ,Granada, 1984.

[10] Colmerauer A., Metamorphosis Grammars, in Natural Language Communication with Computers , ed. Leonard Bolc, pp. 133-191, Springer Verlag, Berlin, 1978.

Statistical Studies in Arabic Linguistic

M. MRAYATI
Scientific Studies and Research Center
P.O. Box 4470
Damascus, Syria

ABSTRACT

This paper deals with the Arabic written text. Statistics on the occurrence of characters and diacritics in words (mazīd) are given. Basic results obtained on the occurrence of character clusters are presented and discussed. Reference will be made to statistics on Arabic roots and words. Some application of these results are in the areas of Arabic printer design, text compression, search algorithms, Arabic character recognition, optimization of diacritic insertion algorithms, spelling correction algorithms etc...

I. INTRODUCTION

The use of natural languages in computer is increasing and is involved in important applications such as office automation. Statistics on natural languages offer models of these languages. These models are used effectively in algorithms to perform or optimize certain operations [1] [2]. One can perform several kinds of statistics depending on the application. The major ones for the Arabic language are the following:

1 - Statistics on the occurrence of consonants in roots.
2 - Statistics on the occurrence of graphemes, digrams and trigrams in written text. Only the graphic shapes of symbols are considered. Different studies of this kind are needed for several applications.
3 - Statistics on the occurrence of phonemes and diphonemes in spoken Arabic.
4 - Statistics on the occurrence of words in Arabic text.
5 - Statistics on the occurrence of Arabic morphological patterns in written text.

Several studies on the Arabic language in the above mentioned fields have been made. Firstly, reference [3] gives a survey of works done on Arabic roots up till 1983 with results on the five main Arabic dictionaries. Secondly an example on a study of the occurrence of characters in Arabic text is given in table (1). This example shows results on the frequency of occurence of characters in Arabic written text using the ASM0449 36 symbols. These results are obtained in Syria, Iraq and Morocco [4] and they are intended to be used in the design or computer keyboard layout. Thirdly, Dr. A.H. Moussa [5] [6] [7] made a study of the frequency of occurrence of phonemes and diphonemes in two samples of the Holy Quran when it is read. He considered 6 vowels and 28 consonants. He found the frequency of occurrence of each phoneme (consonants or vowels). He also gave the frequency of occurrence of the combinations consonants-short vowel (cv) and consonants-long vowel (cv̄). Fourthly, statistics on the occurrence of words in Arabic were performed by several researchers. The work done by Dr. D.A. Abdo [8] is the best that we know of. Fifthly, as far as we know, no work has been done on the statistics of Arabic morphological patterns in written text.

In the following paragraphs, we discuss statistics on written Arabic text from the point of view of shapes of symbols. This study is intended mainly for use in computer peripheral design and in computer Arabic data processing.

II. THE NUMBER OF CHARACTERS IN ARABIC

In reviewing several studies of statistics on Arabic characters, we found that the definition of the set of characters under study is not always clear. This paragraph discusses briefly the different categories of character sets depending on the plane dealt with.

When we are dealing with the phonological plane, the standard Arabic language has 28 consonants and six vowels which are shown in table (2). Sometimes the pharyngealized /l/ is added to make 29 consonants. But since this consonant is very rarely used (ولاه،والله), it can be ignored. We would like to emphasize here, that firstly the vowels are three short ones ـَ, ـُ and ـِ (الفتحة والضمة والكسرة)

and three long ones " ا "," و" and " ي "
(الالف والواو والياء الممدية)
; and secondly, the 3alif " ا "is not a consonant and should not be included with them, while the hamza is a consonant and must always be considered as such. Furthermore, waw 3allin and ya3 3allin
(ي اللينة، و اللينة)
are consonants (corresponding to the semi-vowels W and Y in English), and are different from waw 3almad and ya3 3almad which are long vowels (corresponding to /ū/ and /ī/ in English). Therefore, in Arabic each of the graphemes "و" and "ي" represents two different phonemes: a consonant and a vowel .

When we are dealing with the phonetic plane, each phoneme in standard Arabic could have one or several allophones; example: the pharyngealized vowels. But it is not the purpose of this study to enumerate these allophones.

When we are dealing with the writing or printing of standard Arabic, we should consider the diacritical signs which are used with Arabic characters. There are eleven signs in addition to the three short vowels, namely 14 signs which are shown in table (2). The presence of these signs indicates the following:

1 - The presence of one of the short vowels: ـَ , ـُ , ـِ , namely 3alharakah) /a/ , /u/ and /i/.
2 - The absence of a vowel: ـْ (sukūn).
3 - The gemination of a consonant namely, the presence of two consonants, the first one with sukūn, i.e no short vowel, and the second with one of the three short vowels: ـّ , ـُّ , ـِّ .
4 - The presence of a short vowel followed by an /n/ (tanwīn): ـً ـٍ ـٌ .
5 - The gemination of a consonant followed by "tanwin": ـّـً ـّـٍ ـّـٌ .
6 - The presence hamzat-3alwasl which is pronounced only at the begining of an utterance and ignored elsewhere.

These elevens diacritical signs are used in the writing or orthographic plane, and are intended to transfer the text to the phonetic plane, using definite rules.

On the orthographic or printing plane , standard Arabic has seven additional characters which indicate certain grammatical functions. These are the following:

1 - Graphemes supporting the hamza: أ ، إ ، ى ، و
2 - The tā3 3almarbūtah: ة ;
3 - The 3alif 3almaksurah and the 3alif 3almamdūdah: ى ، آ .

The above mentioned characters and signs form the basic number of graphemes capable of coding Arabic orthographic messages. We summerize the above discussion concerning the total number of graphemes as follows:

28	Consonants
7	Minimum additional set of characters
1	Long vowel (and not three because the two others, و and ي 3almad, have the same shape as the consonants و and ي 3allin).

36	
3	Short vowels
11	Diacritic signs

50	

Finally, when we want to realize the printing of Arabic orthographic message from its coded form (36 characters or symbols without diacriticals and 50 symbols with diacriticals); each character can have different shapes depending on its position in the printed word and depending on the printing style used (Naskh, Thouloth, Koufi...etc). The number of possible shapes can vary from sixty to hundreds of shapes depending on the style. We have proposed a standard Arabic character repertoire composed of 90 shapes plus the diacritics [9].

One can study the existance of five possible shapes for each character, namely: 1) Initial, 2) Medial 3) Medial with no connection to the right (like the initial shape), 4) Final connected and 5) Final isolated. An example of some characters in these five positions follows:

Final isolated	Final connected	Medial	Medial	Initial	
ع	عاد	معرة	ساعد	بيع	باع
ء	-	-	سوءا	-	سوء
ا	السيد	قال	سواد	لها	قالوا
أ	أحمد	سأل	سموأل	يشأ	درأ
إ	إيمان	بإيمان	وإيمان	-	-
آ	آن	مآل	اجراءآت	-	-
و	-	سور	رؤية	لم تسو	امرو
ئ	-	سئل	مائل	منشوء	قارىء

On this plane, the number of shapes of characters to be studied is 36 x 5 = 180 possible shapes.

We have found that, for many computer applications involving Arabic input-output or processing, statistical studies done on the 180 shapes plus the 14 diacritic signs is the best approach to the problem. Statistics on any other plane could be deduced from the above mentioned approach using simple arithmetic. The results of this approach are explained in the next paragraph.

III. STATISTICS REALIZED

We took samples of printed Arabic text from different sources (Journals, newspapers,books....). Diacriticals are fully added. The character set adopted is the 36 characters, plus the fourteen diacritics. The size of the analysed sample is 200 000 characters and diacritic signs. The sukun is included, which means that we have 100 000 character of the set of 36 characters, and 100 000 diacritic signs of the set of 14 signs. The characters are analyzed taking into consideration the five possible positions in a word as explained in the previous paragraph.

Having finished the acquisition of the 200 000 characters and diacritics, we wrote a computer program to produce the desired statistics. The results obtained are presented in the next paragraph. For, convenience, space is included as two spaces, one on the character level (CS) and one on the diacritic level (DS).

IV. RESULTS

Table (3) shows the percentage of occurrence of each of the 36 above mentioned characters plus the 14 diacriticals and the space. From this table we can deduce the following:

1 - Most frequent symbols are:

0(DS), ـَ, ▫(CS), ـّ , ا , ه , ل , ـُ , ي , ص , م , و

2 - The number of words in the analyzed sample is 17533 words; and the total number of characters with the space (without diacritics)is 100 000. Consequently, the average length of a word is 4,70 characters.

3 - The zero entropy is $H0=\log_2(51)=5.67$ and first order entropy

$$H1=\sum_{1}^{51} Pi\ Log(Pi)=4.51$$

and the redundancy

$$R1=1-H1/H0=20.56\ \%$$

On table (4), we present three examples of results obtained in other statistical studies. These examples show that the set of character studied is not always well defined: on the NCC study the long vowel " ا "is included but the " ء "hamza is replaced by a character corresponding to its supporting character, i.e ـئـ - ي - ؤ - و ....etc. The second study on the Holy Quran, the " ﻻ "is included in addition to the " ا "but the"hamza" is also replaced as above. In the third example, the space and the long vowel " ا " is included while the hamza is also replaced. All statistical studies, except those done on phonemes [5] [6] do not distinguish between the consonants و and ي and the long vowels ى and ى .

Table (5) shows our results on the percentage of occurrence of each one of the 36 characters plus the space (i.e., a total of 37 symbols).

$$H0=\text{Log}(37)=5.21\ ,\ H1=\sum_{1}^{37} Pi\ \text{Log}(Pi)=4.34$$

$$R1=1-H1/H0=16.64\ \%$$

Table (6) gives the percentage of occurence of diacritics with respect to each other. We can remark that the diacritics raise the first order redundancy of a text from 16.64% to 20.56% and, of course, much more for the second or third order redundancy.

Table (1) percentage of occurrence of the 36 Arabic characters. Comparison with other results.

SSRC	MAROC	IRAQ	Character
0.38	0.20	0.47	ء
0.15	0.00	0.11	آ
2.38	1.5	1.79	أ
0.12	0.1	0.08	ؤ
1.11	0.1	0.59	إ
0.38	0.4	0.56	ئ
15.00	16.5	11.93	ا
3.59	3.2	4.17	ب
2.73	3.1	4.80	ة
4.42	5.0	4.19	ت
0.66	0.4	0.63	ث
1.25	1.4	1.97	ج
1.79	1.8	2.50	ح
0.84	0.7	1.12	خ
2.72	2.8	3.25	د
0.99	0.8	0.63	ذ
4.17	3.9	6.29	ر
0.51	0.5	1.25	ز
2.16	2.2	3.11	س
0.79	0.9	1.33	ش
0.98	0.9	1.46	ص
0.68	0.5	0.93	ض
0.99	1.1	1.81	ط
0.28	0.2	0.29	ظ
3.49	3.4	3.61	ع
0.44	0.3	0.86	غ
2.75	2.5	2.76	ف
2.28	2.2	3.20	ق
1.93	2.1	1.87	ك
11.80	12.1	8.36	ل
6.01	6.0	6.66	م
5.19	4.9	4.42	ن
3.39	2.9	1.53	ه
5.76	5.6	3.99	و
0.91	0.8	0.71	ى
6.98	7.2	6.76	ي

Type		28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
Consonants	28	ي و هـ ن م ل ك ق ف غ ع ظ ط ض ص ش س ز ر ذ د خ ح ج ث ت ب ء
Vowels	6	ـَ ـُ ـِ ا المد و المد ي المد
Diacritics including the three short vowels	11+3 = 14	ـً ـٌ ـٍ ـْ ـّ ـَّ ـُّ ـِّ ـّ ـّ ـّ
Graphemes minimum set that should be added	7	ء أ إ ؤ ئ ة ى
Character repertoire **	more than 69	example ف ف ف ف ع ع ع ع س س س س ح ح ح ح

Table (2)
Arabic characters at different planes.
* The ّ is usually presented as مَّ
**This repertoire includes the character shapes depending on the character position in the word and on the style used .

ء = .14	ل = 4.91
ا = 6.22	م = 2.44
أ = .98	ن = 2.08
إ = .46	ه = 1.37
آ = .06	ة = 1.14
ب = 1.46	و = 2.37
ت = 1.87	ؤ = .05
ث = .28	ى = .38
ج = .50	ئ = .16
ح = .73	ي = 2.91
خ = .36	ـ = 8.77
د = 1.13	
ذ = .40	ـَ = 12.86
ر = 1.71	ـُ = 7.59
ز = .21	ـِ = 3.01
س = .90	ـْ = 1.67
ش = .33	ـً = .51
ص = .40	ـٌ = .29
ض = .30	ـٍ = .30
ط = .42	ـّ = .43
ظ = .11	ـّ = .15
ع = 1.45	ـّ = .01
غ = .17	ـّ = .02
ف = 1.11	o = 5.36
ق = .94	ـ = 2.76
ك = .78	0 = 15.04

Table (3) The percentage of occurrence of each of the 51 symbols: 36 characters, 13 diacritics and the two spaces. The sample analysed is 200 000 symbols.

ء = .84	ء = .29
ط = .22	ا = 12.45
ع = 2.89	أ = 1.96
غ = .34	إ = .92
ف = 2.22	آ = .13
ق = 1.98	ب = 2.91
ك = 1.56	ت = 3.73
ل = 9.82	ث = .56
م = 4.88	ج = 1.01
ن = 4.16	ح = 1.46
ه = 2.74	خ = .72
ة = 2.28	د = 2.25
و = 4.75	ذ = .81
ؤ = .10	ر = 3.43
ى = .75	ز = .42
ئ = .33	س = 1.81
ي = 5.82	ش = .65
ـ = 17.53	ص = .79
	ض = .59

Table (5)
Percentage of occurrence of each one of the 36 characters plus the space (i.e. 37 symbols) in the text of 100 000 characters.

Table (4)
Some statistics on Arabic consonants plus the long vowel 3alif. و and ي represents both the consonants (و and ي 3allīn) and the long vowels (و and ي 3almad).

N.C.C.- Iraq		The Holy Quran	
Without space		3abu 3alfadā3il 3alma īnī	
% of occurrence	Character	No. of occurrences	Character
18,4	ا	48092	ا
11,6	ل	45109	ن
7,7	ي	33512	ل
6,6	ه	26755	م
6,4	م	25919	ي
6,1	و	25586	و
4,9	ت	16070	ه
4,7	ر	12428	ب
4,3	ن	12246	ر
3,4	د	10528	ك
3,3	ب	9417	ع
3,1	ق	8419	ف
2,6	ف	6213	ق
2,2	ج	5996	س
2,1	س	5978	د
2,0	ع	4939	ذ
1,7	ح	4909	لا
1,7	ع	4322	ج
1,5	ط	4130	ح
1,4	ك	3036	ز
0,9	ص	2505	خ
0,9	خ	2404	ت
0,8	ض	2274	ط
0,7	ش	2111	ش
0,5	ث	2037	ض
0,5	ذ	1672	ص
0,4	ز	1217	غ
0,2	ظ	1105	ث
		842	ظ

Alexandria University - Egypt							
Total		Scientific		Literiture		Humanities	
%	Char.	%	Char.	%	Char.	%	Char.
18,78	SP	17,41	SP	20,30	SP	18,72	SP
15,45	ا	15,45	ا	14,77	ا	16,05	ا
9,22	ل	9,53	ل	8,52	ل	9,54	ل
6,50	ي	6,59	ي	6,62	ي	6,31	ي
5,12	ه	5,15	ه	4,87	ه	5,32	ه
4,77	م	4,86	م	4,66	ن	4,96	م
4,51	و	4,57	و	4,48	م	4,50	و
4,41	ن	4,30	ن	4,45	و	4,28	ن
3,80	ت	4,07	ت	3,45	ت	3,86	ت
3,37	ر	3,42	ر	3,37	ر	3,31	ر
2,82	ب	2,90	ب	2,92	ب	2,81	ع
2,61	ع	2,37	ع	2,61	ع	2,67	ب
2,27	ف	2,30	ف	2,53	ف	2,30	د
2,18	د	2,09	د	2,15	د	2,02	ف
1,78	ق	1,76	ك	1,90	ق	1,90	ق
1,75	س	1,73	س	1,84	ك	1,72	س
1,71	ك	1,70	ع	1,80	س	1,56	ك
1,60	ح	1,51	ق	1,69	ح	1,42	ح
1,15	ج	1,34	ج	1,08	ج	1,02	ج
0,88	ذ	1,06	ذ	0,85	ص	0,80	ط
0,80	ص	0,97	خ	0,85	ذ	0,79	ص
0,80	ط	0,83	ط	0,76	ط	0,76	ش
0,73	خ	0,77	ص	0,76	ش	0,74	ذ
0,72	ش	0,76	ض	0,63	خ	0,61	خ
0,60	ض	0,63	ش	0,56	ض	0,50	ض
0,49	ث	0,59	ث	0,42	ث	0,45	ث
0,46	ز	0,51	ز	0,42	ز	0,40	ز
0,33	غ	0,39	غ	0,35	غ	0,27	غ
0,23	ظ	0,21	ص	0,23	ظ	0,25	ظ

Table (6)
Percentage of occurrence of each sign of diacritics with respect to the other signs.

Percentage	Diacritics	Percentage	Diacritics
.85 =	ّ	25.72 =	ـَ
.30 =	ـً	15.18 =	ـُ
.02 =	ـٌ	6.01 =	ـِ
.04 =	ـٍ	3.34 =	ـْ
10.72 =	o	1.02 =	ـٓ
5.53 =	ـٔ	.59 =	ـٕ
30.07 =	0	.61 =	ـٖ

Table (7) Frequency of occurrence of each character in each of the five positions in a word :
position (1) : Initial
position (2) : Medial position after one of the 23 following characters
ب ت ث ج ح خ س ش ص ض ط ظ ع غ ف ق ك ل م ن ه ي
position (3) : Medial position after one of the 13 following characters
ء ا أ إ آ د ذ ر ز و ؤ ى ة
namely not connected from the right side.

position (4) : Final connected
position (5) : Final isolated
The percentage of occurrence is obtained through dividing by (1000), since the total number of characters in the analysed text without diacritics is 100 000. The set of characters considered is 37.

Total	Final isolated	Final connected	Medial not connected	Medial	Initial	Chr.
286	266	0	20	0	0	ء
12449	371	1555	1676	4216	4631	ا
1961	12	14	193	647	1095	أ
921	0	0	63	301	557	إ
125	0	0	11	58	56	آ
2913	281	277	510	969	876	ب
3734	528	277	432	1757	740	ت
558	62	67	118	251	60	ث
1008	49	35	296	456	172	ج
1464	67	62	294	754	287	ح
724	9	48	243	308	116	خ
2250	199	478	395	1041	137	د
809	31	43	85	520	130	ذ
3429	328	626	620	1715	140	ر
421	35	56	72	244	14	ز
1805	98	81	638	766	222	س
652	10	28	109	372	133	ش
793	24	21	184	474	90	ص
592	67	73	178	233	41	ض
844	37	54	157	515	81	ط
219	6	18	22	135	38	ظ
2891	132	297	365	1358	739	ع
338	8	16	63	178	73	غ
2216	128	170	326	549	1043	ف
1879	105	167	399	954	254	ق
1560	60	260	263	610	367	ك
9816	387	672	6110	1925	722	ل
4879	323	600	684	1911	1361	م
4162	973	1072	683	1198	236	ن
2740	257	680	433	1017	353	ه
2280	436	1844	0	0	0	ة
4746	137	123	544	1761	2181	و
103	3	1	19	80	0	ؤ
752	119	633	0	0	0	ى
327	5	7	253	62	0	ئ
5822	254	1370	971	2640	587	ي
17532	0	0	0	0	0	□

Table (8)For each character, the percentage of occurrence in each position relative to its other positions.

5	4	3	2	1	
93.01	-	6.99	-	-	ء
2.98	12.49	13.46	33.87	37.20	ا
.61	.71	9.84	32.99	55.84	أ
-	-	6.84	32.68	60.48	إ
-	-	8.80	46.40	44.80	آ
9.65	9.51	17.51	33.26	30.07	ب
14.14	7.42	11.57	47.05	19.82	ت
11.11	12.01	21.15	44.98	10.75	ث
4.86	3.47	29.37	45.24	17.06	ج
4.58	4.23	20.08	51.50	19.60	ح
1.24	6.63	33.56	42.54	16.02	خ
8.84	21.24	17.56	46.27	6.09	د
3.83	5.32	10.51	64.28	16.07	ذ
9.57	18.26	18.08	50.01	4.08	ر
8.31	13.30	17.10	57.96	3.33	ز
5.43	4.49	35.35	42.44	12.30	س
1.53	4.29	16.72	57.06	20.40	ش
3.03	2.65	23.20	59.77	11.35	ص
11.32	12.33	30.07	39.36	6.93	ض
4.38	6.40	18.60	61.02	9.60	ط
2.74	8.22	10.05	61.64	17.35	ظ
4.57	10.27	12.63	46.97	25.56	ع
2.37	4.73	18.64	52.66	21.60	غ
5.78	7.67	14.71	24.77	47.07	ف
5.59	8.89	21.23	50.77	13.52	ق
3.85	16.67	16.86	39.10	23.53	ك
3.94	6.85	62.25	19.61	7.36	ل
6.62	12.30	14.02	39.17	27.90	م
23.38	25.76	16.41	28.78	5.67	ن
9.38	24.82	15.80	37.12	12.88	ه
19.12	80.88	-	-	-	ة
2.89	2.59	11.46	37.10	45.95	و
2.91	.97	18.45	77.67	-	ؤ
15.82	84.18	-	-	-	ى
1.53	2.14	77.37	18.96	-	ئ
4.36	23.53	16.68	45.35	10.08	ي

Table (9) The percentage of occurrence of each diacritic sign after each one of the 36 characters these and the space are 100 000 couples of character-diacritic(CD).

- Means that the percentage =0

~ Means that the percentage <.005

ٰ	ٖ	ٗ	٘	ٙ	ٚ	ٛ	ٜ	ٝ	ٞ	ٟ	٠	٩	٥	
.07	.12	.04	-	-	-	.01	.03	.01	-	-	-	-	-	ء
-	-	-	-	-	-	.42	-	-	-	-	-	5.53	6.51	ا
1.69	~	.17	-	-	-	~	~	-	-	-	.09	-	-	أ
-	.92	-	-	-	-	-	-	-	-	-	-	-	-	إ
-	-	-	-	-	-	-	-	-	-	-	-	-	.13	آ
.95	1.34	.22	.05	.02	.01	-	.05	.01	~	-	.28	-	-	ب
1.57	1.05	.34	.36	.05	.01	-	.09	.02	-	-	.25	-	-	ت
.18	.14	.07	.08	.01	.01	-	~	~	-	-	.06	-	-	ث
.50	.17	.12	.01	.01	~	-	~	-	-	-	.20	-	-	ج
.76	.24	.12	.01	~	.01	-	.02	.01	-	-	.31	-	-	ح
.25	.15	.08	.01	~	-	-	-	~	-	-	.22	-	-	خ
.83	.47	.29	.15	.11	.08	-	.05	.02	-	~	.26	-	-	د
.43	.25	.04	.01	.01	~	-	.01	~	-	-	.06	-	-	ذ
1.38	1.04	.38	.12	.06	.05	-	.06	.03	~	-	.31	-	-	ر
.12	.10	.04	.04	.01	.01	-	.01	~	-	-	.08	-	-	ز
.59	.35	.12	.12	.05	.05	-	.01	~	-	-	.53	-	-	س
.30	.08	.04	.10	.01	.02	-	~	-	-	~	.10	-	-	ش
.22	.13	.10	.09	.06	.02	-	~	-	~	~	.17	-	-	ص
.24	.18	.06	.03	~	~	-	.01	~	-	-	.08	-	-	ض
.36	.15	.06	.12	.01	.02	-	.01	~	-	-	.12	-	-	ط
.10	.06	.03	.02	~	.01	-	-	~	-	-	.01	-	-	ظ
1.63	.46	.26	.01	~	-	-	.03	.01	-	-	.48	-	-	ع
.20	.05	.04	~	-	-	-	-	-	-	-	.05	-	-	غ
.68	1.14	.13	.03	.02	.01	-	.02	~	-	-	.20	-	-	ف
.97	.40	.24	.03	.01	.01	-	.03	.01	-	~	.19	-	-	ق
.84	.24	.25	.03	.02	.01	-	.01	~	-	~	.17	-	-	ك
2.10	1.40	.36	.44	.10	.09	-	.06	.02	~	~	3.69	-	1.54	ل
1.90	1.24	.96	.16	.05	.03	-	.04	.03	~	~	.47	-	-	م
1.50	.71	.24	.45	.05	.03	-	.03	.01	-	~	1.15	-	-	ن
1.11	.86	.60	.01	~	~	-	~	~	-	-	.16	-	-	ه
.20	1.25	.27	-	-	-	.17	.29	.11	-	-	-	-	-	ة
2.90	.19	.05	.09	.05	.03	-	~	-	~	~	.47	-	.96	و
.04	-	.03	-	-	-	-	-	~	-	-	.03	-	-	ؤ
-	-	-	-	-	-	.01	-	-	-	-	-	-	.75	ى
.04	.27	~	-	-	-	-	~	~	-	-	.01	-	-	ئ
1.07	.04	.27	.80	.31	.10	-	~	-	.01	.03	.54	-	2.66	ي
-	-	-	-	-	-	-	-	-	-	-	-	-	17.53	□

Table (10) The frequency of occurrence of the trigram: diacritic-character-diacritic/DCD/, in a text of 200 000 graphemes with 36 characters and 14 diacritics.

- Means that the percentage =0
~ Means that the percentage <.05

ء	آ	أ	ؤ	إ	ئ	ا	ب	ة	ت	ث	ج	ح	خ	د	ذ	ر	ز	س	ش	ص	ض	ط	ظ	ع	غ	ف	ق	ك	ل	م	ن	ه	و	ى	ي	□	
5.3	3.1	1.1	.9	.3	.2	.5	.2	.1	~	~	4.1	.6	9.2	ء
2.8	1.0	.5	.9	.3	.1	-	.1	~	~	~	1.4	.2	7.9	آ
1.0	.2	.5	.1	.1	.1	-	~	~	~	~	.8	-	3.1	أ
.6	.9	.3	~	~	~	.1	.1	~	-	-	.4	-	.9	ؤ
.3	~	.1	~	~	~	-	~	~	-	~	~	-	.5	إ
.1	.1	.1	~	~	-	-	~	~	-	-	.1	-	.2	ئ
-	-	-	-	-	-	-	-	-	-	-	-	-	.6	ا
-	-	-	-	-	-	-	-	-	-	-	-	-	.9	ب
-	-	-	-	-	-	-	-	-	-	-	-	-	.3	ة
-	-	-	-	-	-	-	-	-	-	-	-	-	~	ت
-	-	-	-	-	-	-	-	-	-	-	-	-	~	ث
4.8	2.8	1.4	-	-	-	-	.1	~	-	-	-	.1	1.4	ج
-	-	-	.2	~	-	-	-	-	-	-	3.8	-	1.5	ح
10.9	7.1	1.9	1.1	.3	.3	~	.3	.1	~	~	~	4.6	3.5	خ

Table (12) The list of the characters that did not come after each one of the 36 characters in the analysed sample of 100 000 characters.

ء	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
آ	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
أ	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ؤ	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
إ	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ئ	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ا	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ب	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ة	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ت	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ث	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ج	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ح	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
خ	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
د	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ذ	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ر	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ز	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
س	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ش	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ص	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ض	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ط	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ظ	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ع	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
غ	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ف	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ق	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ك	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ل	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
م	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ن	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ه	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
و	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ى	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
ي	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □
□	ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ى ي □

Table (11) The frequency of occurrence of the digram: character-character (CC) in a text of 100 000 character with space.

	ء	ا	أ	إ	آ	ب	ت	ث	ج	ح	خ	د	ذ	ر	ز	س	ش
ء	0	10	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
ا	251	1	4	0	0	237	567	52	116	137	58	241	20	418	49	343	31
أ	0	0	0	0	0	73	22	33	38	56	104	71	8	63	23	141	36
إ	0	0	0	0	0	22	1	4	15	33	8	12	68	5	2	60	17
آ	0	0	0	0	0	3	3	8	0	0	28	14	0	1	0	2	0
ب	0	396	52	9	0	25	71	7	7	92	11	75	22	119	4	27	29
ت	0	185	38	0	3	119	90	8	61	155	89	51	10	152	17	68	54
ث	0	65	0	0	0	14	3	0	0	0	0	2	0	45	0	0	0
ج	0	179	6	0	0	35	67	0	0	6	0	60	14	48	28	13	1
ح	0	141	0	0	0	37	104	29	15	1	0	163	12	122	31	39	3
خ	0	74	0	0	0	31	50	0	1	0	3	55	35	80	7	5	4
د	1	271	14	0	0	50	58	49	1	3	22	48	0	129	0	8	0
ذ	0	228	0	0	0	17	9	0	3	0	0	0	0	19	0	0	0
ر	43	473	5	0	8	233	78	3	71	57	9	68	0	12	17	59	5
ز	0	58	1	0	0	24	9	0	3	4	0	8	0	17	3	0	0
س	0	305	5	0	0	97	225	0	10	9	5	25	0	90	0	6	0
ش	0	119	19	0	1	24	24	0	8	0	9	13	0	103	0	0	0
ص	0	124	0	0	0	43	1	0	22	2	0	53	0	110	0	0	0
ض	0	106	0	0	0	6	6	0	15	3	0	9	0	46	0	0	0
ط	0	149	6	0	0	74	4	0	6	0	0	1	0	113	0	3	0
ظ	0	31	0	0	0	0	0	0	0	0	0	0	0	53	0	0	0
ع	0	283	0	0	0	89	74	11	28	0	0	169	0	232	20	8	16
غ	0	33	0	0	0	6	8	0	0	0	0	9	4	51	5	0	9
ف	0	145	10	32	0	4	75	0	18	16	4	5	5	101	10	72	4
ق	0	253	0	0	0	59	76	0	0	2	0	256	1	66	7	24	13
ك	0	266	9	1	2	50	108	52	0	1	0	7	43	97	7	13	9
ل	781	484	259	51	3	283	539	100	123	316	106	164	100	110	46	247	127
م	687	19	0	0	0	36	195	64	116	88	46	53	5	193	29	142	49
ن	440	1	0	1	0	53	147	2	25	33	4	53	28	3	23	107	32
ه	637	0	0	0	0	24	28	0	19	0	0	97	261	72	15	1	0
و	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ى	999	143	62	3	0	132	192	26	98	71	23	121	20	283	13	119	29
ي	7	7	0	0	0	0	19	5	0	0	0	11	0	1	0	4	1
ة	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ى	11	0	0	0	0	0	14	8	0	3	0	17	6	13	3	1	0
ي	361	12	0	0	0	131	125	44	51	51	74	182	13	322	48	74	47
□	4631	1095	557	0	56	876	740	60	172	287	116	137	130	140	14	222	133
	12449	1961	921	125	3	2913	3734	557	1008	1464	724	2250	809	3429	421	1805	652

	ص	ض	ط	ظ	ع	غ	ف	ق	ك	ل	م	ن	ه	و	ى	ي	ة	ى	ي	□	
ء	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ا	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
أ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
إ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
آ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ب	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ت	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ث	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ج	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ح	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
خ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
د	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ذ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ر	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ز	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
س	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ش	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ص	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ض	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ط	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ظ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ع	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
غ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ف	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ق	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ك	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ل	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
م	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ن	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ه	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
و	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ى	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ي	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ة	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ى	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ي	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
□	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	592	793	652	1805	421	3429	809	2250	724	1464	1008	557	3734	2913	125	921	1961	12449	286		

	ة	ي	ئ	ى	ؤ	و	ۀ	ه	ن	م	ل	ك	ق	ف	غ	ع	ظ	ط	
286	266	1	0	0	0	0	2	4	1	0	0	0	0	0	0	0	0	0	ء
12449	1926	95	225	0	9	77	42	156	627	360	5487	74	219	161	21	206	20	84	ا
1961	26	117	0	5	0	194	3	46	412	172	76	33	27	30	6	44	1	19	ب
921	0	20	0	0	0	0	0	1	245	20	285	0	9	9	3	54	0	12	ت
125	0	5	0	0	0	3	0	2	19	5	26	1	0	4	0	0	0	0	ث
2912	558	488	2	1	1	54	66	149	119	63	129	34	58	9	9	164	1	30	ج
3734	805	203	2	34	9	125	15	247	141	219	142	74	165	79	32	157	8	81	ح
558	129	61	0	0	0	25	24	12	21	46	65	1	36	0	9	0	0	0	خ
1008	84	56	2	0	0	53	28	85	32	96	80	0	0	5	4	26	0	0	د
1464	129	151	0	1	0	80	34	14	14	75	78	31	81	12	0	0	19	5	ذ
724	57	53	0	0	0	21	5	10	7	7	110	0	0	2	0	0	0	65	ر
2250	677	197	6	26	1	204	139	79	26	116	37	12	22	29	0	25	0	0	ز
809	74	96	2	1	0	17	4	145	8	1	158	26	0	0	0	1	0	0	س
3429	954	386	20	56	12	120	206	111	33	32	0	92	54	89	12	31	0	4	ش
421	91	50	3	0	0	31	17	12	16	19	18	0	10	3	1	15	0	0	ص
1805	179	193	1	1	13	101	50	54	27	99	145	20	18	23	0	45	0	59	ض
652	38	52	2	3	3	15	5	33	4	22	3	45	12	16	1	74	0	4	ط
793	45	61	0	1	0	98	21	19	26	14	76	0	3	41	3	18	0	6	ظ
592	140	63	0	4	0	21	11	22	1	27	40	0	0	5	3	58	0	5	ع
844	91	65	4	4	1	100	17	20	42	17	52	0	29	16	3	21	0	6	غ
219	24	21	0	1	0	10	6	20	1	28	23	0	0	1	0	0	0	0	ف
2891	429	156	0	6	0	50	103	64	220	106	607	9	51	8	0	0	14	14	ق
338	24	75	0	3	0	21	34	2	6	20	20	0	0	5	0	0	0	1	ك
2216	298	839	6	4	1	32	84	68	30	15	74	53	62	8	1	79	9	8	ل
1879	272	169	0	7	0	158	149	30	17	12	101	0	7	47	0	40	3	57	م
1560	320	102	1	0	0	64	35	11	97	111	135	0	1	12	0	5	0	0	ن
9816	1059	442	10	529	2	337	178	242	195	944	282	378	262	207	107	507	17	134	ه
4879	923	234	14	7	35	163	158	107	740	80	162	78	60	38	11	268	2	18	ة
4162	2045	298	2	25	0	103	62	255	9	40	1	16	53	108	0	29	77	46	و
2740	937	115	0	2	6	136	27	7	40	262	41	8	1	0	0	0	0	0	ۀ
2280	2280	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ؤ
4746	260	254	2	31	0	19	23	129	269	277	393	81	163	129	28	121	7	75	ى
103	4	4	0	0	0	16	0	5	0	5	17	4	0	0	0	0	0	0	ئ
752	752	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ي
327	12	76	0	0	0	0	22	22	12	27	46	7	4	1	1	10	0	0	ة
5822	1624	37	23	0	10	117	710	204	469	181	185	116	218	76	10	154	3	30	ي
17532	0	587	0	0	0	2181	0	353	236	1361	722	367	254	1043	73	739	38	81	ة
	17532	5822	327	752	103	4746	2280	2740	4162	4879	9816	1560	1879	2216	338	2891	219	844	

Table (7) Shows the results obtained on the occurrence of each of the 36 characters plus the space in each possible position in a word . These results take the space into consideration . But if the application needs these results without the space, each number in the table should be scaled by a factor which is 100 000/(100 000-17532). We give, in **table (8)**, for each character, its percentage of occurrence in each of tha five possible positions or shapes. For example , the hamza " ء " comes 93.01% of the time in final isolated position but never comes in initial or medial connected position. The character "ة" comes 80.88% of the time in final connectec position and 19.12% in final isolated and never comes elsewhere. This table could be used to optimize Arabic character recognition systems.

Table (9) Shows the frequency of occurrence of diacritic signs after each character in a text of 200 000 graphemes.

Table (10) Shows the results obtained on the percentage of occurrence of the trigram: diacritic-character-diacritic /DCD/. The text analysed is composed of 200 000 graphemes. The set of characters is the 36 characters plus the 14 diacritic signs.

Finally, **table (11)** gives the frequency of occurrence of all the digrams, character-character. The text analysed is 100 000 charaters. The set of characters is 37 including the space. To get the percentage of occurrence of each digram, one should divide by 1000. For example the percentage of occurrence of the digram "ال" is 5.487% and that of the digram "لم" is 0.944%

$$Ho = \log_2 (37) = 5.21$$

The second order entropy is $H2=\sum Pij \; Log(Pij)=3.6$

and the redundancy is R2=1-H2/2H0=65.3 %

Finally from table (11) we can deduce **table (12)** which shows the digrams which did not come in our analysed sample.

V. APPLICATIONS

Possible uses of the results obtained in this study are enumerated below with references:

1 - Design of computer keyboard layout [9] . Table (3) and results on occurrences of the digrams 49 x 49 are used in addition to the spaces.
2 - Design of Arabic daisy wheel [8] . Table (3) , (7,8) and similar results could be used.
3 - Character recognition in context [2].
4 - Continuous speech recognition [1].
5 - Text compretion , spelling error detection, and dictionary compretion [10].
6 - Linguistic studies: for example comparison with the results obtained on the roots [3] and conclusion on the relation between these two results.

ACKNOWLEDGEMENT

The author would to thank Mr. M. Bawab for his closed collaboration throughout this study; and also Mr. Y. Mer Alam and Mr. H. Tayan for the fruitful discussion that he had with them.

REFERENCES

[1] LALIT, R.B. and al, "A Maximum likelihood approach to continuous speech recognition". IEEE Trans. on Pattern Analysis and Machine,Intelligence, Vol. PAMI-5, No. 2, March 1983.

[2] THOMAS, R.B. and Kassler, M. "Character Recognition in Context"; Information and Control, 10, PP. 43-64, 1967.

[3] MRAYATI M., "Statistical Studies of Arabic Language Roots" Proceeding of Arab School on Science and Technology. Rabat Sep. 1983.

[4] ASMO files

[5] MOUSA, A.H. "Computer Application to Arabic Roots and Arabic Words". Proccedings of the Arab School on Science and Technology; Rabat, 1983.

[6] الدكتور علي حلمي موسى "استخدام الالات الحاسبة الالكترونية في دراسة ألفاظ القرآن الكريم" مجلة عالم الفكر، المجلد ١٢ صفحة ١٥٣-١٩٤ ، ١٩٨٢ .

[7] MOUSA, A.H."Occurrence Probability of Characters and Entropy of Arabic Language " Proceedings of ASST; Zabadani Vally, Syria 1985.

[8] الدكتور داوود عطية عبده. "المفردات الشائعة في اللغة العربية" مطبوعات جامعة الرياض١٩٧٩ .

[9] MRAYATI, M., and MOUNAJED, B., "Design Criteria of Arabic Peripherals", Proceeding of the Arab School on Science and Technology., Zabadani, July 1985.

[10] IDLEBI, N., and MRAYATI, M., "Design of Arabic Keyboard Layout, based on Statistical properties of Arabic Characters", Proceedings of Arab School on Science

[11] McMAHON L.E., CHERRY L.L. and MORRIS R., "Statistical Text Processing" BSTJ, Vol, 57, No.6, July, August 1978.

Morphology and Syntax of the Arabic Language

YAHYA HLAL
E. M. I.
B.P. 765
Rabat, Morocco

1 Introduction

1.1 The stages of analysis.

It has become customary, in the automatic processing of natural languages, to consider four stages:

- Morphological processing, which associates features of various sorts with words, taking no account of context.
- Syntatic processing, which associates syntactic structures with phrases, taking into consideration the respective positions of words.
- Semantic processing, which aims at removing the ambiguities remaining after syntactic processing, by reference to the semantic relations which bind concepts together.
- Pragmatic processing, which places the phrase in the context of the general realm of knowledge in order to remove the ambiguities which cannot be eliminated by semantic processing.

1.2 The importance of morphological analysis in Arabic.

Among the problems posed in Arabic is that of identifying the lexical word in a text in order to associate it with the information contained in a dictionary. This problem arises out of the fact that certain prepositions and conjunctions, the definite article, various pronominal forms, and some other particles are closely bound to the lexical word as prefixes or suffixes. This means that a single lexical word may appear in text in more than a hundred different forms. To resolve this difficulty, it is necessary to decompose such forms into primary morphological elements.

$$w \rightarrow \sum P + (R, Sch) + \sum S$$

where:

$\sum P$: prefixed elements

$\sum S$: suffixed elements

R: radical

Sch: inflectional schema

example

و بمدرستهم <-- و+ ب + (درس + مفعلة) + هم

1.3 The objective of morphological analysis.

The objective of morphological analysis is to associate with the word:

- its elements after decompositon (outside of context$\Longrightarrow$ eventual ambiguities)
- grammatical and semantic information (exclusive of context)

example

فهم <-- + (فهم ، فعل) + فَهِمَ فَهِمْ ...
و+ (هم ، فعل) + فَهَمْ...
و + هم

One can see, in this example, that فهم , out of context, can carry the sense of "understanding" فهم , or "solicitude" همم , or represent merely a combination of conjunction and pronoun.

1.4 The objective of syntactic analysis

The objective of syntactic analysis is to associate with the phrase being processed the appropriate syntactic tree (or trees), which will allow for the elimination of the ambiguities inherent in a context-free morphological analysis, and will identify phrase segments which have syntactic functions.

example فهم ولد الرجل الكتاب المفيد

This phrase is made up of three segments: one verbal (فهم), and two nominal, each of the latter having a different syntactic

function.

فاعل : (ولد الرجل)

مفعول به : (الكتاب المفيد)

This may be represented in parenthesized notation, thus:

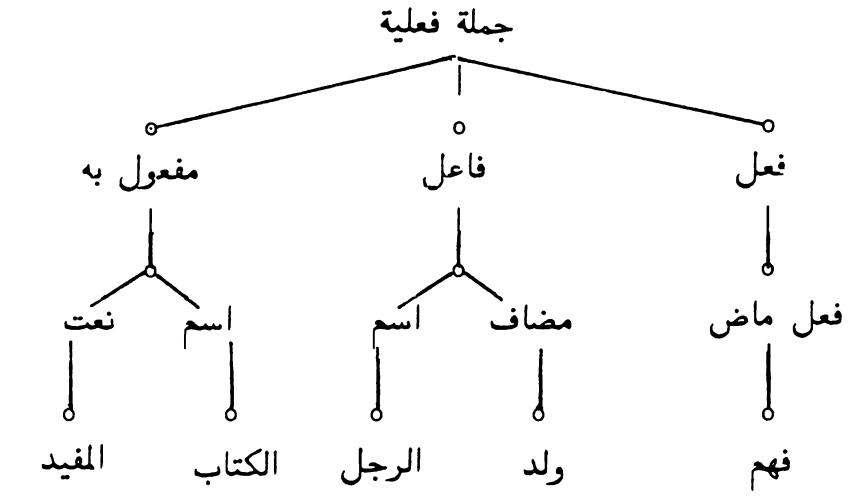
((فهم) (ولد الرجل) (الكتاب المفيد)

| | |

فعل فاعل مفعول به

جمله فعلية

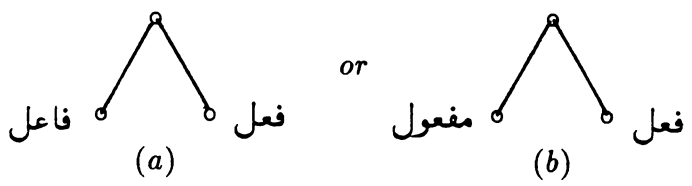
or in a more expressive way by a tree structure.



Remark: At the syntactic level, excluding any consideration of semantic content, it is possible to associate more than one syntactic tree with a given phrase, which will result in several different interpretations of that phrase. This frequently occurs in unvowelled Arabic. For example.

أكلت السماكة

This could be either:



In (b), there remains the further uncertainty whether أكلاتْ or أكلاتَ or أكاتْ is intended.

2 The morphological analysis system

The system gives different treatment to:

- the class of regular words (nouns, verbs, ...), for which it is possible to achieve a decomposition into *radical* and *schema.* مدرسة --> (درس - مفعلة)

  دارس --> (درس - فاعل)

  مدروس --> (درس - مفعول)

  يدرس --> (درس - يفعل)

- the class of function-words (prepositions, conjunctions ...).

On

- analysis of function-words.
- analysis of regular words.

Note that a given letter-sequence may be any of the following:

- function-word (في)
- regular word (مدرسة)
- (context-free) combination of function-word and regular word. (فهم)

2.1 Definitions

1. primary function-words: function words which are not in composition with other words (من ، على ، فى). Note that فيهم and وعليهم cannot be considered primary.
2. the microgrammatical class (CAT) function-words: the label (CATW) is given to the set of function-words which are identified through the processing applied to them as playing a common rôle. We can define the microgrammatical class 'prefix' ($CATP$) and 'suffix' ($CATS$) in a similar manner.
3. the set of context-free gramatical values (CFV) associated with a word: i. e., the entire set of grammatical values which can be assumed by a word according to its function in a text.

2.2 Processing of function-words

This processing depends on the existence of the following three elements:

- dictionary of primary function-words.
- set of rules for the validation of decompositions.
- decomposition algorithm.

1. dictionary of function-words

$$< \text{word} \longleftarrow CATW >$$

2. set of rules for validation. The rules take the following form:

$$\underbrace{CATW_1, CATW_2, \ldots, CATW_n}_{\text{name of the rule}} : \underbrace{(CFV, A)}_{\substack{\text{validation} \\ \text{(unvowelled} \\ \text{word)}}} \mid \underbrace{\sum (VW \rightarrow CFV, A)}_{\substack{\text{validation} \\ \text{(vowelled word)}}}$$

- the names of the rules consist of a series of microgram-

- (CFV, A) is the validation for an unvowelled element, where:

  CFV: CFV to be associated with the element after decomposition.

  A: a boolean value, indicating whether this word is to be processed as a regular word.

- $\sum(VW \rightarrow CFV, A)$ is the validation for a vowelled element. VW represents the set of vowelled forms associated with this word. This aspect of the rule permits an additional stage of validation by reference to the vowelled form. The non-vowelled form, after decomposition, will be rejected if the rule is not applicable to the vowelled form.

  This rule may be read as follows:

  If the decomposition of an unvowelled word or a vowelled word (VW) stripped of its vowels produces a series of primary function-words $W_1, W_2, \ldots, W_n$, belonging to the classes $CATW_1, CATW_2, \ldots, CATW_n$, then the decomposition is valid, and the word as analysed has for its CFV the set of context-free grammatical values associated with that series. If the word is vowelled, the validation holds true only if the rule is applicable to the vowelled form.

3. decomposition algorithm.

   Processing is carried out as follows:

   (a) Extraction of the unvowelled consonantal form if the word is vowelled.

   $$WORD \rightarrow CWORD, VWORD$$

   (b) An attempt is made to decompose the unvowelled form $(CWORD)$ into primary function-words by repeated reference to the dictionary of function-words. Two results are possible.

   i. The word is not composed exclusively of primary function-words; in this case, if no further decomposition is possible, the residuum is subject to analysis by the procedure for regular words.

   ii. The word is composed exclusively of function-words; in this case it will be necessary to make certain that the decomposition is valid. For this, the sequence of grammatical categories must correspond with rules established in the grammar. If so, and if the original word was not vowelled, the decomposition is declared valid.
   If the original word was vowelled $(VWORD)$, the rules indicated must be applicable to the vowelled form. (See the interpretation of validation rules above.)

2.3 Processing of regular words.

Working hypothesis.

The system proposed here depends on the following working hypothesis, which has been confirmed by *a posteriori* results.

"Knowledge of the longest prefix (P) and the longest suffix (S) in a word (W) together with the length of the presumed radical (R) which would result from this *a priori* decomposition

$$W \rightarrow P + R + S \qquad LR = \text{Length}(R)$$

is sufficient to lead to a correct analysis of regular words."

This hypothesis is at the basis of the morphological rules used in our system.

A morphological rule has the following form:

$$\underbrace{(CATP, CATS, LR)}_{\text{argument}} \rightarrow \underbrace{\sum A}_{\substack{\text{sequence of} \\ \text{actions to be} \\ \text{undertaken}}}$$

This rule is applied to the class of regular words whose longest prefix and suffix belong to the microgrammatical class CATP and CATS, and whose radical length LR is the length of the sequence of consonants which results from the decomposition into prefix, suffix and presumed radical (R). $\sum$A is the series of actions to be undertaken with the aim of achieving definitive results in morphological analysis.

Dictionaries.

- To construct the argument of a rule, we must have available dictionaries of prefixes and suffixes arranged so that it is possible to derive the longest possible sequences from them as well as the constituent elements from which these sequences are composed.

$$< \text{prefix} \rightarrow CATP, CFVP >; < \text{suffix} \rightarrow CATS, CFVS >$$

where CFVP consists of the context-free values (CFV) conveyed by the elements.

- The actions to be undertaken require the use of a dictionary of strong radicals (through which the existence of any given radical may be verified), dictionaries of defective radicals so that these may be evaluated in sequence, and finally a dictionary of *schemata* to make the information conveyed by these last elements available.

Module for the analysis of regular words.

This module is in effect a structure for the implementation of morphological rules, which has the following advantages for the wywtem as a whole

- ease in updating the module, since it is is governed only by the repertory of basic actions used by the rules, and not by the rules themselves. This makes the module quasi-independent of the difficulties involved in the development

- the morphological rules deal with classes of words in such a way that the elaboration of a rule is achieved by simply considering the class of words affected by that rule, independent of the language involved; there is therefore an intellectual benefit derived from the elaboration of the rule.
- the system is portable, because its essential elements are dictionaries.

Actions used in writing the rules.

Reference is made to [3] [5], where will be found the necessary details concerning the syntax used for writing rules, along with the functional description of each action. Here we will merely enumerate the actions.

The actions required for the constitution of the rules are divisible into 8 categories.

1. Action on the radical part: modification of a character (M), suppression of a character (S), doubling of a character (D), transferrence of a prefix(/suffix) or part of one into the radical (T).
2. Action on the infix region: insertion of an infix (X).
3. Evaluation of elements in the radical: recognition of a character in the radical part including possible infixes (C), recognition of a sequence of characters (R) (this requires the use of a dictionary of radicals, weak radicals and other elements), recognition of any incompatibilities between radical and schema (Q), recognition of the length of an element (K).
4. Validation of prefixes and suffixes (V).
5. Association of information carried by these (I).
6. Completion or suppression of a decomposition (A, B).
7. In case of failure at any of the above stages, listing of the elements as found.
8. Actions and evaluations on the vowelled region (necessary only if the text is vowelled).

Steps in the analysis of regular words.

1. Determination, by consultation of dictionaries of prefixes and suffixes, of the key of the morphological rule to be applied (determination of $CATP$ and $CATS$) and calculation of the length of the remainder R which results from an *a priori* decomposition.
2. Invocation of the rule, using the key established in the previous step.
3. Execution of the actions required by the rule.

Syn

- to determine that prefixes and suffixes are compatible.
- to complete the synthesis of the schema, so as to permit access to the information (gender, number, etc.,) conveyed by each element.
- to determine case or mood (nominative, genitive, accusative, subjunctive, jussive) whenever this is possible (especially if the word is vowelled).
- to determine the information conveyed by the radical (transitive or intransitive, lexico-semantic relations).

3 Syntactical analysis

In general, the problem is to make evident the structure (or, in cases of ambiguity, the structures) of the phrase being analysed. This structure may be represented by a tree, in which each node represents a syntactic function.

Each structure represents a particular instance of a given type of phrase. The set of definitions of structures associated with a language constitutes the grammar to be associated with that language.

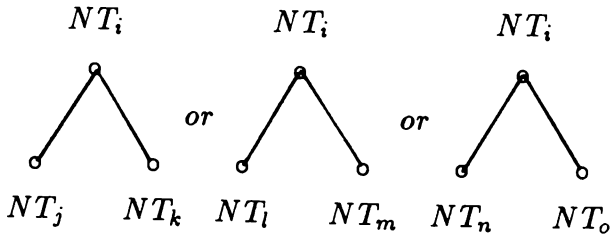
In a formal sense, a grammar is defined by the following quadruplet:

$$G_L =< V_T, V_N, P, S >$$

V_T : terminal vocabulary, i.e. the set of grammatical values
V_N : non-terminal vocabulary, i.e. the set of syntactical functions
P : the set of production rules describing the structure of phrases in the language

$$NTi \rightarrow < NTj >< NTk > | < NTl > [T_W] | [T_P]\,[T_P] \ldots$$

NT : non-terminal T : terminal This rule describes the following structures:



S : axiom, i.e. the particular non-terminal from which the tree is to be developed.

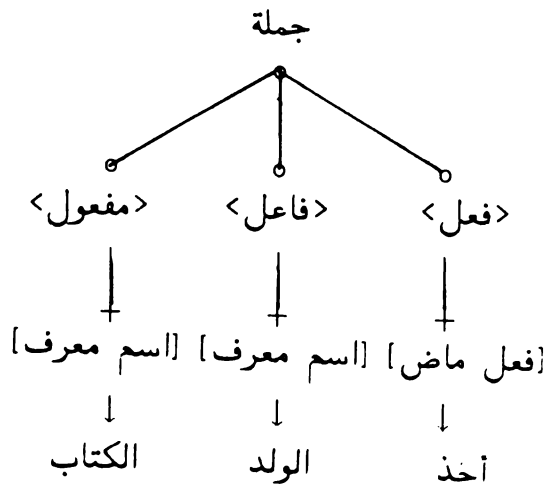
Example: To describe a phrase of the type

أخذ الولد الكتاب

the production rules are:

جملة --> <فعل><فاعل><مفعول>	
<فعل> --> [فعل ماض]	1
<فاعل> --> [اسم معرف]	2
<مفعول> --> [اسم معرف]	3
	4

The stage of morphological analysis associates the grammatical اسم معرف ، فعل ماض and اسم معرف with أخذ ، الولد respectively. The grammar then permits the generation of the following structure.



This automatic generation proceeds as follows:

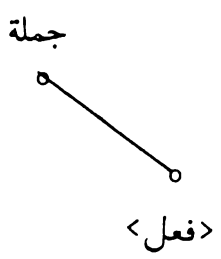
Starting from the axiomatic جملة one applies rule No. 1.

< مفعول >< فاعل >< فعل >

⇑ AS = 1.

AS

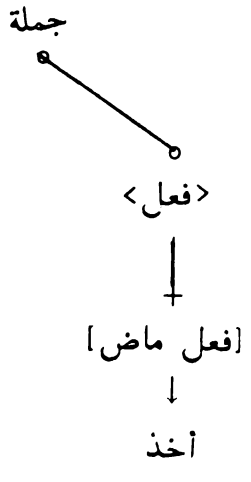
Construction of the part



< فعل > allows one to apply rule No. 2.

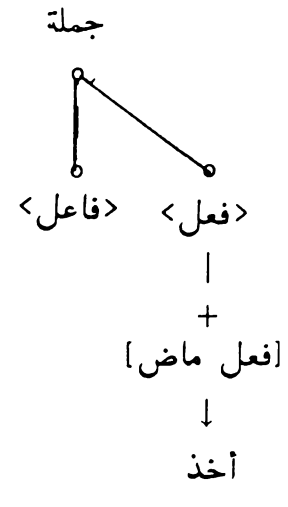
< مفعول >	< فاعل >	< فعل >	$AS = 1$	Rule 1
-1	-1	[فعل ماض]	$AS = 1$	Rule 2

[فعل ماض] ($AS = 1$) is a terminal. At this point it is necessary to verify that the actual word (أخذ) has this value. It does, and therefore that branch of the tree is put in place before passing to the evaluation of the next word



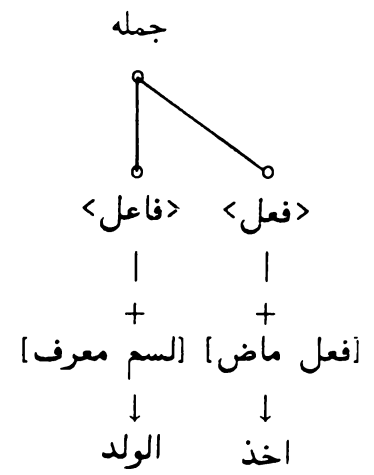
Now we pass on to the next part of the rule $AS = 2$, which is empty (this rule contains only one part). In this case it is necessary to go back to the previous rule

Here the referenced part $AS = 2$ does exist. It must therefore be put in place before going on to apply the rule which corresponds to it (Rule 3).



< مفعول >	< فاعل >	< فعل >	$AS = 2$	Rule 1
-1	-1	اسم معرف	$AS = 1$	Rule 3

At this point we begin the same operations as were associated with Rule 2.

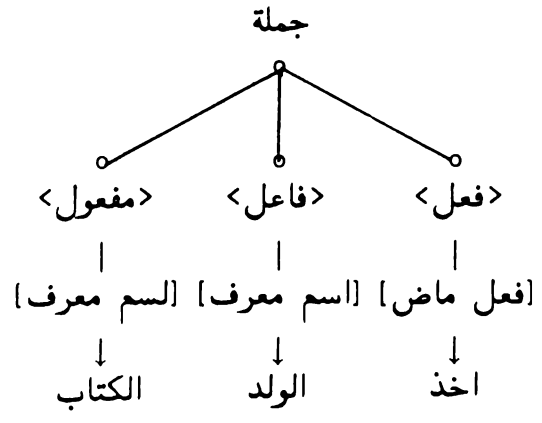


The fact that we again reach the value for no entry (-1) requires that we go back once again to Rule 1.

< مفعول به >	< فاعل >	< فعل >	$AS = 3$	Rule 1

The corresponding part of the rule exists, so we put it in place before going on to apply the rule that corresponds with it.

< مفعول به >	< فاعل >	< فعل >	$AS = 3$	Rule 1
-1	-1	اسم معرف	$AS = 1$	Rule 4



3.1 Alternatives provided in the rules.

$$P \rightarrow < A > | < B >$$

The rules can be extended to provide for the inclusion of phrases. For example, rule 2 can be extended to include phrases of the type:

سياخذ الولد الكتاب او سوف ياخذ الولد الكتاب

2′ <فعل>--><فعل ماض>|<فعل مضارع>

[س+ فعل مضارع] | [سوف] + [فعل مضارع]

3.2 Recursiveness in rules

$$P \rightarrow < A > | < A > < B >$$

Nominal parts are of the type

$^{m}($ نعت $)$ اسم $($ اسم مضاف $)^{n}$

for example ولد رجل مدينة البلاد الجميل

correspond to $n = 3$ and $m = 1$. The description of these parts can be achieved by recursive rules. Such parts can have every one of the functions compatible with the characteristics of nouns (فاعل ، مفعول ...). Rules 3 and 4 can therefore be modified to accommodate these types of parts.

		$< SN >$ ← < فاعل >		3′
		$< SN >$ ← < مفعول >		4′
$< SAD >$	[اسم]	$< SA >$ ←	$< SA >$	5
$< SA >$	[مضاف]	[مضاف] ←	$< SN >$	6
$< SAD >$	[نعت]	[نعت] ←	$< SAD >$	7

3.3 The null alternative.

$$P \rightarrow < A > | \ \Phi$$

The null alternative permits us to stipulate that a part may be not appear explicitly in a structure. For example, the part <فاعل> in أخذ الكتاب . The same holds true for the part <مفعول> in أكل الولدُ . In these cases rules 3′ and 4′ can be completed as follows:

Φ	$< SN >$ ←	< فاعل >		3″
Φ	$< SN >$ ←	< مفعول >		4″

to reflect these situations.

3.4 The notion of a rule with predicates

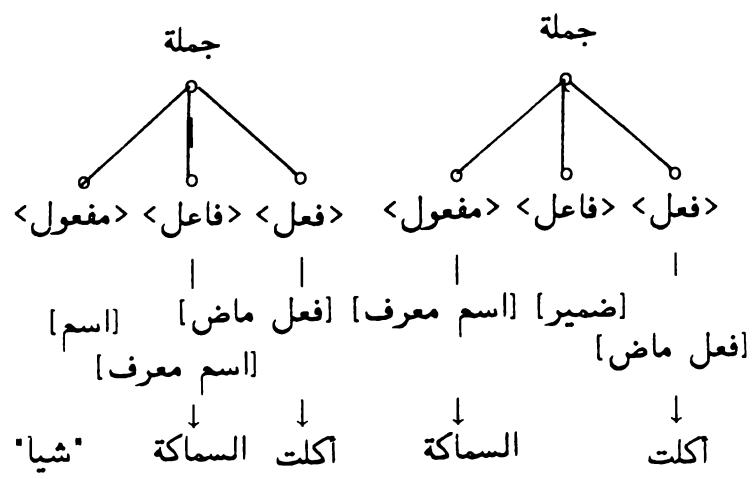
In the case of rules which offer different alternatives, the problem posed to choose among them. In these cases it is necessary to indicate what data can lead to a choice. Predicates associated with rules are logical expressions bound to each alternative. Thus, the value of a predicate determines whether to chooose the alternative in question. A rule with a predicate has the following form:

$$A \rightarrow (P_1) < B_1 > | \ (P_2) < B_2 > | \ \ldots (P_n) < B_n >$$

which states that A is rewritten as B_1 if P_1 is true, as B_2 if P_2 is true and B_n if P_n is true.

In principle, $P_i \wedge P_j$ is false for all $[i, j]$, but in cases where this in formal languages, therefore, is to avoid "back-tracking" and achieve a more rapid treatment of syntax.

The use of predicates allows us the possibility of gaining a general idea of the nature of the sentence, which offers a guide to its syntactic structure as the evaluation of words proceeds. To put it another way, predicates make it possible to avoid "back-tracking" from the level of the chosen aslternative. In the case of Arabic, rules with predicates prove very useful for elucidating certain structures and removing ambiguity in certain cases. For example, the following sentence, " أكلت السماكة ," when written without vowels, leads to the construction of two syntactic trees, reflecting an ambiguity—one does not know whether the fish *ate* or *was eaten.*



If the sentence is even partially vowelled, in such a way as to indicate the grammatical case of the noun " " (nominative or accusative) then the choice of the correct structure is possible. It is necessary only to modify rules 3′ and 4′ as follows:

Φ	$< SN >$ (case = ^{U} ←	< فاعل >		3′
Φ	$< SN >$ (case = ^{A} ←	< مفعول >		4′

which stipulates that the syntactic function < فاعل > is not to be selected unless the actual word is in the nominative (case = ^{U} : < حالة الرفع >) and that < مفعول > is not to be selected unless the word is in the accusative (حالة النضب) This does no more than formalize some very elementary concepts.

A second example shows the difficulty of defining adequate structures. Let us take the sentences:

أكل خبزاً ، أكل أكلاً

On would find it difficult to distinguish *a priori* the syntactic function of خبزاً (مفعول به) from that of أكلاً (مفعول مطلق)). *A posteriori,* what makes the choice possible is the consideration of radicals, which we shall formulate as:

$$J * = 1$$

which stipulates that the radical of the current word $(*)$ is to be compared with the radical of the word in position 1. It is necessary, in fact, to be even more general, and to be able to write that the radical of the current word is to be compared with the radical of the verb in any earlier position.

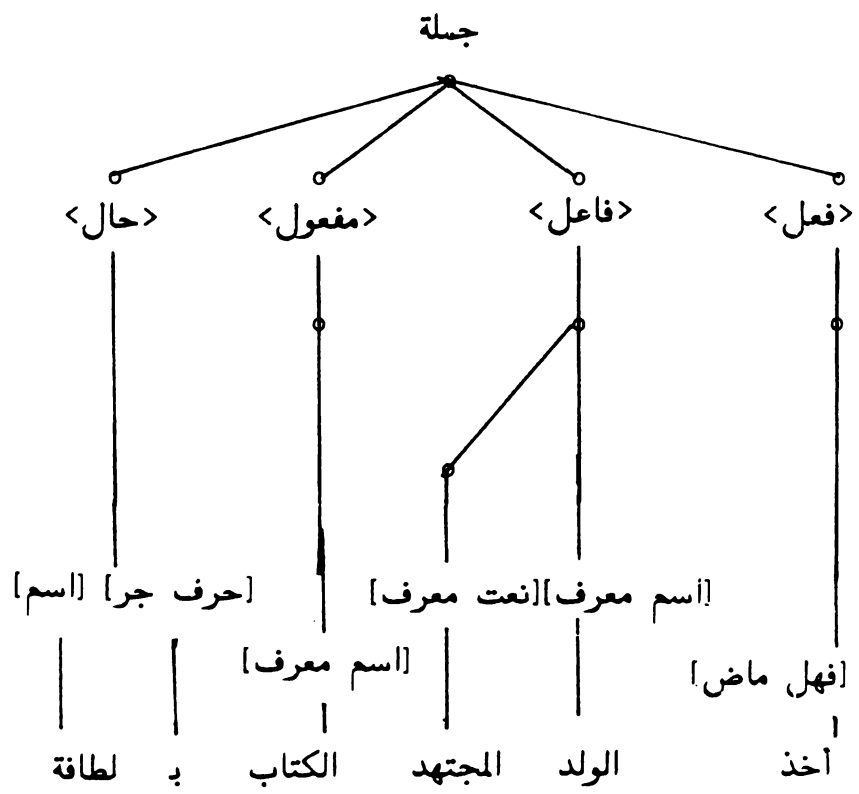
The predicate is a logical expression describing a grammatical situation which one attempts to identify in order to choose the associated alternative.

As an illustration, we give here some elements of possible situations:

K	1	=	مَنْ	(K for كلمة)
K	E	=	'?'	(End of sentence = '?')
N	1	=	verb	(N for نَعْو, the grammatical value of word 1 is 'verb')
J	i	=	ز	(J for جذر)
S	i	=	ال	(S for سابق)
L	i	=	وما	(L for لاحق)
@	*	=	• - ١	(@ for عدد , agreement in number)
G	*	=	• - ١	(G for جنس , agreement in gender)
C	*	=	'A'	(K for الإعراب , grammacase.)
etc.			...	

4 Conclusion.

The syntactic structures established by a system of rules of this type are capable of permitting the interpretation of sentences. Thus, in: أخذ الولد المجتهد الكتاب بلطافة



we can ask the following questions.

Who (مَنْ) took the book?

We can verify, by following the tree, that the word "book" exists, functioning as a direct object, and that "took" exists, functioning as a verb, and in this case the answer is the word which functions as a subject < فاعل >, which is to say الولد . Thus the question "who" implies the function subject < فاعل >.

What did the boy take? (ما ذا)

<مفعول> <= ما ذا

<حال> <= كيف

etc....

We see that this permits a level of interpretation compatible with applications such as computer-aided instruction, automated indexing, and data-base query systems (systems allowing access to a data-base through Arabic language queries which can be equated with formal queries) etc.

In those cases where predicates are bound to vocalization, and the vowels are not written, as in أكل الولد , it will be necessary to resort to semantic processing, since we cannot tell from syntax alone whether the boy is subject or object. In a case such as this, semantic processing will probably indicate that the boy is more likely to eat than to be eaten.

References

[1] C. Fluhr *Algorithmes à apprentissage et traitement automatique des langues.* Theèse d'etat, Université de Paris XI. June 1977.

[2] Y. Hlal *Méthodes d'apprentissage pour l'analyse morphosyntaxique (expérimentées dans le cas de l'arabe et du français).* Thèse d'etat, Université de Paris XI. Feb. 1979.

[3] Y. Hlal *Analyse morphologique de l'arabe non voyellé* B. E. O. Tome XXXIV, 1982.

[4] Y. Hlal *Système imformatique et l'arabe.* "Arab School of Science and Technology." Rabat, 1983.

[5] Y. Hlal *Traitement morphologique de l'arabe.* (in Arabic.) Kuwait, 1985

A Comprehensive Arabic Morphological Analyser Generator

BOTROUS THALOUTH
IBM, Kuwait Scientific Center (KSC)
P.O. Box 4175
Safat, Kuwait

ABDULLAH AL-DANNAN
College of Education
Kuwait University, Kuwait

1. INTRODUCTION

1.1. The Problem

Computer Scientists and applied linguists have been recently paying big efforts to develop useful algorithms in all fields of computational linguistics. In English, French and some other foreign languages successes have been reported in language analysis, translation and generation. However, long work should be devoted before all problems in this concern will be solved.

The need for the cooperation within multidisciplinary teams of linguists and computer scientists, is a well recognized fact. Kuwait Scientific Center has been working in this direction. In fact the research reported here is the result of joint work between Botrous Thalouth from the Kuwait Scientific Center, as computer scientist and Dr. Abdullah Al Dannan, applied linguist of Kuwait University seconded during the 1984-1985 Academic year to the Kuwait Institute for Scientific Research (KISR). The approach and algorithms presented here are believed to have a solid linguistics base, with an algorithmic implementation which combines simplicity and flexibility.

A number of papers dealing with morphological analysis of the Arabic language have been published in a recent workshop entitled "Computer Processing of the Arabic Language" held in Kuwait April 14-16, 1985. Only one paper, presented by Y. Hlal [1] gave detailed examples to demonstrate the workability of morphological decomposition. Hlal based his analysis on the assumption that the triple (class of largest size prefix, class of largest size suffix, number of letters remaining in the word), characterizes, in the most efficient way, the set of rules to be applied to find the proper decomposition.

Our approach is basically different. It is characterized by the extensive use of pattern matching and flags to implement differences in morphological rules. Moreover, prefixes and suffixes are defined in such a way that their interference with patterns is minimized.

1.2. The Aim

This research aims at:

* Developing a comprehensive Arabic morphological analyser and generator (CAMAG), with the following considerations. For the CAMAG to be practical, orthography used should be identical with that prevailing in Arabic newspapers, magazines, and most books, i.e. void of diacritics (short vowel signs). Also, the algorithm should be compact, concise, and easy to handle.

* Building up a lexical data base for Arabic dependent on CAMAG, and which can be used as a reference for categorization, translation and other educational and cultural purposes.

1.3. Approach

The above aims were achieved through three stages:

1. Morphological analysis of modern Arabic words according to certain hypothesized algorithms in an attempt to pinpoint as many problems as can be found.

2. Linguistic investigations in Arabic in order to predict areas of difficulties which might face CAMAG, in

addition to those spotted in stage 1.

3. Development and implementation of algorithms of CAMAG which overcome all predicted difficulties, and work smoothly and effectively.

2. HYPOTHESIZED ALGORITHMS FOR DECOMPOSITION OF MODERN ARABIC WORDS

2.1. Description of Words used

Words for analysis were taken from large Arabic computerized data sets, representing a total of 3 million words. The most frequent 3 thousand words, covering 93% of the data sets were selected.

2.2. Hypothesized Algorithms used

The aim, at this stage, was to investigate the practical problems which need to be taken in consideration for the following steps. Thus, the hypothesized algorithms were simple: stripping the most frequent suffixes and prefixes and then matching the word with most frequent patterns so as to get the triliteral origin of the word.

2.3. Types of Problems

The examination of the analysed words, revealed a large number of problems which were studied and classified. Following are the most outstanding types.

* Some words were not analysed because they belonged to patterns not included in the algorithm, e.g. أسلوب

* Words whose initial or final radical letters were identical with suffixes or prefixes were given wrong triliteral origin, e.g. قاضى was given the origin قـاض instead of قضى, العاب was given the origin عاب instead of لعب.

* Some prefixes were not omitted because they were considered radical letters, e.g. أدار was given the origin أدر instead دار

* Letters in the root which were changed into other letters in certain patterns caused problems, e.g. انحـاد was given the origin تحـد instead of وحـد.

* Words which lost one of their radical letters were wrongly analysed, e.g يرضـون was given the origin يرضـ instead of رضـي

* Demonstrative words such as هؤلاء, relative words such as الذيـن were also wrongly analyzed.

Obviously, the texts started with did not represent all texts in Arabic. Consequently, the list of problems was not exclusive of all the possible ones. It was therefore inevitable to search for a systematic way in order to :

* Predict all possible difficulties.

* Develop the algorithms which will always perform the correct word analysis and generation.

Investigating the Arabic language: its lexis, grammar, and semantics in the light of computational needs and demands seemed to be the necessary step to be taken first.

3. LINGUISTIC INVESTIGATIONS IN ARABIC

3.1. Preliminary Notes

These investigations are concerned more with morphology than the other aspects of the Arabic language systems. Phonology, Syntax, and semantics were dealt with as much as they clarify or add to the morphological analysis.

Classifications and definitions of linguistic elements were intended to serve the mechanism and cause of Arabic computational linguistics. In other words, we did not completely follow labels and definitions of Arab grammarians. For example, certain words labelled as nouns by Arab grammarians do not necessarily bear the same label in our descriptions of Arabic. Also, in certain patterns we used morphological computational balances (MCB) which Arab grammarians completely refused and prohibited [2]. For example, we used افدعـل instead of افتعل to balance ازدهـر .

For computational purposes Arabic lexis can be classified into two groups. Structural words and content words. Structural words are closed sets and content words are open sets. All members of sets are liable to be attached to prefixes and/or suffixes.

3.2. Structural Words (Closed sets)

Structural words are those which are used to help in building up the structure of the sentence. They function as: noun substitutes (pronouns, demonstratives, ...etc), linkers (relatives, prepositions) and interrogatives ...etc. In short, they are formally distinguished by being closed sets.

By closed sets we mean groups of words that will never increase in number, i.e. no new member will be added to the group.

Some of the groups defined under this category do not necessarily correspond with the grammarian classification, e.g. ليس، عسى etc...

The total number of the members of the closed sets is 199. Some members of certain groups can be attached to suffixes, e.g. the adverb بعد or the preposition الى become اليه، بعدها، بعده اليها . Others do not accept suffixes such as: لن ، هذا.

It was found that the closed sets and all their combinations with suffixes were less than 500.

3.3. Content words (open sets)

By content words we mean words which have a dictionary meaning. Foreign words imported into Arabic fall under this heading.

Open sets are those liable to increase in order to express new meanings. They can be classified into : Verb forms and noun forms.

1. Verb forms

Verb forms are patterns of verbs and their derivatives. Verb patterns are inflected for tense, mood, person, number and gender. Some of them undergo changes in the passive voice and when they are preceded by a "nasb" or "jazm" particle.

The other patterns derived from verbs are :

a. Abstract noun المصدر

b. The mould of the agent اسم الفاعل

c. The mould of the patient اسم المفعول

d. Adjective moulds used as agent الصفة المشبهة

e. The mould of prominence أفعل التفضيل

f. Moulds of exaggeration صيغ المبالغة

g. Noun of one action مصدر المرّة

h. Noun of kind مصدر الهيئة

i. The noun of "Mimi" المصدر الميمي

j. The artificial noun المصدر الصناعي

k. The noun of instrument اسم الآلة

l. Noun of time اسم الزمان

m. Noun of place اسم المكان

Derivation rules for the categories above can be stated:

* Categories: (a),(g),(h),(i),(j),(k),(l), and(m) are inflected for number.

* Categories: (b),(c),(d),(e) and (f) are inflected for number and gender. They are also inflected for case, i.e. their dual and masculine plural suffixes depend on whether they are in the accusative or nominative case.

* All derivatives mentioned above can be derived from the triliteral verbs.

* Categories (d),(e),(f),(j) and (k) are not derived from verbs with more than three letters.

* Some derivatives of the types (b) and (c) have broken plural جمع تكسير in addition to their "safe plural" جمع سالم .

* All categories from (a) to (m) have diminutive patterns تصغير.

2. Noun Forms (solid nouns)

Noun forms are patterns of solid nouns أسماء جامدة and other nouns derived from them. The basic patterns for these nouns can be triliteral like فرس, quadriliteral like حصان or pentaliteral like سفرجل

Derivation rules of noun patterns are :

* All nouns are inflected for number, some for gender.

* Singular noun patterns do not follow a regular rule in forming their plurals. Two words of the

same pattern may have two different plural patterns called "broken plurals" جموع تكسير , e.g فرس and جبل are pluralized أفراس and جبال respectively.

* All nouns may have derived verb forms, e.g. تشمّس.

3.4. Suffixes and Prefixes

Grammarians agree that an affix (suffix or prefix) is a syllable or a sound attached to the end (or beginning) of a word to add a meaning.

Under our definition, prefixes and suffixes are syllables which must be removed to prepare the word for pattern matching. For example, يفعل does not contain any prefix as far as computer analysis is concerned because يفعل is one of the stored patterns in the CAMAG algorithm. At the same time, the conjunction is treated as a prefix.

These definitions meet the requirements of algorithmic analysis and generation but do not comply with conventional grammarian labels.

3.5. Predictions of Problematic Areas:

The classes defined below were conjugated to cover all possible computational morphological balances (CMB).

* Strong unglottalized verb, السالم الصحيح
* Strong initially glottalized verb, مهموز الأول
* Strong medially glottalized verb, مهموز الوسط
* Strong finally glottalized verb, مهموز الآخر
* Strong double lettered verb, المضعف
* Weak verb: first radical, المثال
* Weak verb: second radical, الأجوف
* Weak verb: third radical, الناقص
* Weak verb: first radical, and third radical اللفيف المفروق
* Weak verb: second radical, third radical اللفيف المقرون
* Noun patterns representing different forms with consonant and vowel radicals are stated, e.g. رجل ، رجال فتية ، فتى ..etc.

Then a systematic classification of problems to be solved in the decomposition of words was undertaken. These are:

* When the first radical is ن the pattern انفعل is liable to be confused with افتعل, e.g. انتصر may be analysed as تصر
* When the first radical is س the problem is two fold:
  - س may be confused with the prefix س, and in this case it will be automatically removed.
  - the pattern افتعل , e.g استمع will be confused with the pattern استفعل , e.g. استجدّ .
* Verbs whose first radical is ز or ذ will be confusing when they occur in the pattern افتعل because both are changed into د .
* Verbs whose first radicals are ص, ض, ظ might also be wrongly analysed when they occur in the pattern افتعل because their first radical is changed into ط .
* When the first radical is a glottal stop "hamza" ء , eg أمن the pattern إفعال will be confusing because إيمان will be identical with ايراد and ايناع of the same pattern but with different first radicals ي and و respectively.
* When the second radical is a glottal stop the pattern فاعل will be confusing because سائل "questioner" of سأل is identical with سائل "liquid" of سال .
* When the final radical is a glottal stop the pattern استفعال will be confusing because استدعاء from دعا , and استيفاء from وفى , and استملاء from ملاء might be wrongly given the same origin.
* The present tense form يجد of the double lettered verb المضعف can be confused with the present tense after لم from وجد , جاد , أجوى respectively. All of them are written لم يجد .
* Words that end with the suffixes أن , ين , ين , cause problems in words like أمان , تمرين because they might be analysed with wrong patterns.
* Words that begin with ف , ف , ب , ك , ل , ال, create problems because all these are removed as prefixes.

* The patterns افتعل , انفعل of the double lettered verb are confusing, e.g. امتد , انمد might be analysed متد and نمد respectively.

* When the first radical is و the patterns إفعال and استفعال are problematic, e.g. إيراد and استيراد might be wrongly analysed as يرد instead of ورد.

* When the second radical is و the patterns مستفعل , مفعله and أفعل (passive) will create problems, e.g. مقالة , مستقيل , أقيل. In this case قال or قيل will be wrongly given instead of قول

* When the second radical is ي the patterns افتعل and استعمل are problematic, e.g. ابتاع and استباع will be wrongly analysed as باع instead of بيع

* When the first and the third radicals are و and ى respectively the imperative will be one letter, e.g وقنا is و/ق/نا. This word might be wrongly analysed as وقن or قنا.

* When the third radical is ى the patterns يفاعل and فاعل might cause problems in certain contexts, e.g. لم يقاض , هذا قاض. Such words might be wrongly analysed as قاض instead of قضى.

* Foreign words pose problems because they do not fit in the morphological pattern system of Arabic. At the same time they can be preceded and/or followed by certain prefixes, e.g تركي. When the suffix ي is removed the remaining word ترك is not the correct answer.

* Quadriliteral and pentaliteral solid nouns الأسماء الجامدة can be confusing when they end or begin with letters identical with a suffix or a prefix, e.g لسان , سلاح , دكان.

4. APPROACH TO LEXICAL DATA BASE IMPLEMENTATION

The concept of building a lexical data base system, using a set of small size lexicons of roots, in conjunction with morphological decomposition and generation rules is well known. The challenge is to develop a proper implementation, based on a solid linguistics base.

The previous paragraphs describe the linguistic approach adopted, which classifies all Arabic words as structural words and content words. Content words are in their turn exhaustively classified in a number of morphological classes, with new Arabic patterns created to accommodate all the special cases.

From an implementation point of view, the requirement is that the definitions and modifications of the morphological classes be totally separated from the program logic. Thus the system remains open ended even when it reaches maturity. This approach was adopted both in the word decomposition and generation phases.

The extensive use of morphological classes greatly simplifies both the decomposition and generation algorithms. This is especially true in the case of generation algorithms, which are simply implemented through flags associated to the root entries (pointing to a set of accepted patterns for that root).

Another interesting aspect is the use of generation rules in the decomposition algorithms of content words (to validate the results of the decomposition), as well as a tool to build the lexical data base.

The algorithms described below use logically separate lexicons for prefixes, suffixes, verb roots, solid word roots, foreign words and structural words.

5. MORPHOLOGICAL DECOMPOSITION ALGORITHMS

* The algorithm seeks valid decompositions of a given word, first as a structural word, then as a derived content word, then as a foreign word.

5.1. Structural Word Analysis

The total number of structural words, stripped from their prefix (و , بـ etc..) was evaluated to be less than 500. Consequently the analysis of structural words simply consists of stripping them from their prefix, and consulting the corresponding lexicon. The steps of the algorithm are:

1. Separate the largest prefix found in the prefix table for the word being analyzed.

2. Check if the remaining part of the word is in the structural words lexicon.

3. If a structural word is identified, check a number of flags and perform corresponding actions.

   * A flag specifying a class of accepted prefixes will determine if the decomposition is accepted.

   * A flag specifying if the structural word should also be analyzed as a content word.

   * A flag specifying if the structural word is compound. It then also specifies the location of the components boundaries.

   * A flag specifying the grammatical value of the structural word (for none compound structural words).

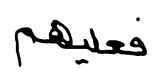
The last 2 flags provide syntax information for further processing.

Examples

Ex 1.

* Extract the prefix و.
*  belongs to structural words lexicon.
* The structural word هم accepts the prefix و (flag 1).
* وهم can also be a content word (flag 2). Therefore, it will also be analyzed through the content word analysis.
* Since flag 3 is not on (simple structural word), flag 4 will give its grammatical value as ضمير منفصل لجمع المذكر الغائب.

Ex 2.



* Extract the prefix ف.
*  belongs to structural words lexicon.
* The structural word عليهم accepts the prefix ف (flag 1).
* consists of two structural words على and هم (flag 3).

5.2. Content Word Analysis

The steps for content word analysis are described below. It is of course implied that the prefix has already been stripped from the word during structural word analysis. Furthermore, the term "roots lexicon" in the algorithm refers to both verb roots lexicon and solid words lexicon, which are scanned in that sequence when a root is being searched.

1. Separate the largest suffix found in the suffix table from the word being analyzed.

2. Perform a preliminary check on the compatibility between the prefix and suffix, using flags in the prefix and suffix tables. Words which fail this test will perform step No. 8.

3. Match the remaining part of the word with patterns classified into groups according to their length.

4. For every matched pattern, check a number of flags to eliminate unacceptable patterns.

   * A flag determines the prefixes compatible with the pattern, with a similar flag for suffixes. Compatibility between prefixes and suffixes has already been checked in a previous step.

   * A flag specifies if any (very) special constraints exist with the pattern. For example a flag in the pattern افدعال causes a check on the first radical of the root, which must be ن or د.

5. Determine the possible roots associated to the word

   * For most patterns, the root is determined directly by pattern matching.

   * Some patterns have a flag which determines a special procedure to get the root. There are only 2 different procedures, each with 3 alternative actions.

6. Check the existence of each possible root in the roots lexicon.

7. For each possible root which is matched in the lexicon, check (through a flag in the roots lexi-

con) if the original pattern is among the accepted patterns generated from this root.

8. To insure that no possible decomposition has been missed, steps 3 to 7 are repeated, first with the addition of the suffix, then with increasing parts of the prefix (starting with the inner prefix), then with the increasing parts of the prefix, without the suffix.

Examples.

1. Words with strong roots matched with one pattern:

وبالمعلومات

* The prefix found in the prefix table is وبال.
* The suffix found in the suffix table is ات
* Pattern matching on مفعول gives the root علم.
* The root lexicon entry علم is flagged to indicate that the pattern مفعول is among its derivatives.

2. Words with strong roots matched with more than one pattern:

المنتصر

* The prefix found in the prefix table is ال.
* There is no suffix.
* Two patterns مفتعل and منفعل can be matched with منتصر.
* The pattern منفعل is flagged to indicate that the letter matching the position of ف in the root pattern may not be a ت. It is noteworthy that the same flag applies to a number of other patterns such as انفعال and انفعل

3. Words where some root characters have been modified:

a. Words where a character of the pattern is replaced by another:

ازدهار

* No prefixes or suffixes are detected.
* One pattern افدعال can be matched.
* This pattern is flagged to indicate that the letter matching the position of the ف in the root pattern is ز or ذ. The pattern is accepted.
* Pattern matching on افدعال gives the root زهر.
* The root lexicon entry زهر is flagged to indicate that the pattern افدعال is among its derivatives. In fact this flag simply classifies the root in the appropriate group.

b. Words where a character of the root is replaced by another:

Ex 1.

استيراد

* No prefixes or suffixes are detected
* Two patterns استيعال and استفعال can be matched.
* There are no apriori constraints (flags) on these patterns.
* An action flag exists on استيعال to add a و in the first position و of the root pattern.
* The two roots determined by pattern matching are then ورد and يرد.
* Among the 2 possible roots ورد and يرد associated with the above patterns, only ورد appears in the root lexicon.
* The flag associated with the root ورد in the roots lexicon indicated that استيعال is one of the generated patterns.

Ex.2.

قال

* The pattern فـال can be matched.
* There are no apriori constraints on this pattern
* An action flag indicates that either the letter و or يـ has to be placed in the ع position of the root pattern.
* Of the 2 resulting possible roots, only قول is found in the lexicon, with the proper flag defining فال among the acceptable patterns.

c. Words where one character of the root is missing:

اتصل

* No prefixes or suffixes are detected.
* Two patterns اتعل and افعل can be matched.
* There are no apriori constraints (flags) on these patterns.
* An action flag exists on اتعل to add a و or ا in the first position ف of the root pattern.
* Among the 3 possible roots وصل and اصل and تصل associated with the above patterns, only وصل appears in the root lexicon.
* The flag associated with the root وصل in the roots lexicon indicates that اتصل is one of the generated patterns.

d. Words where two characters of the root are missing:

وقنا

* The prefix found in the prefix table is و.
* The suffix found in the suffix table is نا.
* One pattern ع can be matched.
* An action flag exists on to add a و in the first position ف, and ى in the last position ل of the root pattern.
* The flag associated with the root وقى in the roots lexicon indicates that وقى is one of the generated patterns.

4. Words whose ending or beginning are identical with prefix or suffix letters:

Ex.1.

المعين

* The prefix found in the prefix table is ال.
* The suffix found in the suffix table is ين.
* The pattern فع can be matched
* The roots extracted in this way are not found in the lexicon
* The suffix is then appended to give معين.
* The pattern مفيل can be matched.
* The root عين is obtained and checked in the lexicon etc..

Ex.2.

فألاعبهم

* The prefix found in the prefix table is فال.
* The suffix found in the suffix table is هم.
* The two patterns فعل and أفع can be matched.
* Neither of the 2 patterns leads to any valid root entry in the lexicon.
* The suffix هم is then appended to give اعبهم.
* This direction also fails to produce any accepted word.

* A new trial is then done with the first inner prefix appended, and fails.

* Finally, the suffix is extracted, and a pattern matching with the inner prefix gives the pattern أفاعل.

* This will yield the right root لعب.

5.3. Foreign Word Analysis

This analysis is systematically invoked after content word analysis. It is therefore implied that the prefix and suffix have been already extracted. The analysis then consists of the following steps.

1. Check the existence of the word in the foreign word lexicon.

2. Check (through a flag in the foreign word lexicon), if the prefix and suffix are accepted prefixes and suffixes.

3. If step 2 fails and the suffix is not equal to يا or ة steps 1 to 2 are repeated, first with the addition of the suffix ي, then with the suffix ة.

4. If step 3 fails, steps 1 to 3 are repeated, first with the addition of the suffix, then with increasing parts of the prefix (starting with the inner prefix), then with the increasing parts of the prefix, without the suffix.

Examples.

Ex 1.

الصينيين

* The prefix found in the prefix table is الـ.

* The suffix found in the suffix table is يين

* The word صين is in foreign word lexicon.

* The foreign lexicon entry صين is flagged to indicate that the prefix الـ and the suffix يين are accepted.

Ex 2.

ولبيبيريا

* The prefix found in the prefix table is ول.

* The suffix found in the suffix table is يا.

* The word يبير is not in foreign word lexicon.

* The suffix يا is then appended to give يبيريا

* This direction also fails to produce any accepted word.

* The inner prefix is then appended to give ليبيريا.

* This will yield the right word ليبيريا.

6. MORPHOLOGICAL GENERATION ALGORITHM.

The generation rules are algorithmically quite simple. They are practically developed in 2 steps.

1. Initially the roots are classified according to the classes mentioned above, (see 3.5). Each class defines the complete set of all (morphologically) possible patterns associated with it.

   A user interactive facility was developed to generate and manipulate all the patterns associated to a given root. Given a root inputted by the user, it will first automatically identify its morphological class. Then it will generate all the possible patterns of that root, using the patterns defined by its morphological class.

   These patterns are classified in a rational way according to the usual Arabic classification of grammatical values.

2. The user (the linguist who develops the lexical data base) browses through these patterns, and marks the unused ones. The system will use this information as an input to perform a final classification of the roots (according to the actually used patterns) and will flag the roots according to this new classification.

The lexical data base (for words actually used) built in this way is the

one used for validation in the decomposition algorithm.

7. CONCLUSION

Working in a multidisciplinary team has been a challenging, exciting and rewarding experience.

The approach of starting with heuristic algorithms with a practical set of data, to discover the real problems to be solved, then establishing a comprehensive linguistic framework that solves those problems, turned out to be quite effective.

The algorithms described have been implemented and are being thoroughly tested. Results are quite encouraging.

A root based comprehensive lexical data base for Arabic can now be built. Besides efficiency in terms of storage and retrieval, the built-in morphological analysis can also provide syntactic information which can be used in syntax parsing.

ACKNOWLEDGMENT

The authors like to express their gratefulness to Dr. Samer Attasi, manager of Kuwait Scientific Center, for his encouragement, help, and valuable comments.

REFERENCES

1. Hlal Y. تحليل صرفي للعربية , Proc. Computer Processing of Arabic Language, Kuwait , Vol 1, 1985.

2. جامع الدروس العربية ، الشيخ مصطفى الغلاييني المكتبة العصرية ، بيروت ، ١٩٨٢ .

From a Free Arabic Text to Its Word Frequency Table: The Search for a Solution

ABDIN M. ABDELKARIM
Khartoum International Institute For Arabic Language

ABSTRACT

Extracting the word frequency table of free linguistic text-for possible use in some statistical inferences-is an easy matter in languages whose traditions of transcription code vowels and avoid attaching meaningful linguistic entities such as prepositions, definite article etc to beginings and ends of words. Arabic is not among those languages. Vowelisation in Arabic is still to be generalised in a standard from and affixes are there to stay.

This short paper explores this problem and attempts some solutions.

1. INTRODUCTION:

A computer programme used for extracting word frecuency tables from free texts is usually a three-pass programme. The first is a text conditioning pass in which the lines of the text are formatted to a word per line format suiting the constraints of input to the second pass. The second pass is a sorting pass the outcome of which is an elphabetically ordered list of the text words. What is left is for the third and final pass to output the word and its frequency of occurence. The latter is done by a simple count of the number of times the word is repeated down the ordered list of pass two.

2. THE PROBLEM

With some languages the words of a free text could be separated from each other in a word per line format by using the normal puncuation characters as delimiters. Such a convenience is offered by Languages like English where vowels are coded and affixes are limited to semantic and not transcription purposes. With Arabic where vowelization is rare and affixes of non-semantic bearings are common the entity of the individual words may be obscured. If only punctuation characters are used as separators between Arabic words the resulting separation will not be between actual individual words but between written clusters of words which may need further separations if actual Arabic words is the aim. Obviously if no further separation is performed then words like رجل , الرجل و كرجل will each have its own separate count and take its own different place in the word frequency table. A scheme capable of shedding such affixes as attached to the word رجل is therefore needed at the text conditioning stage if the words رجل are to be fully counted irrespective of their affixes.

Those were the problems emerging at the text conditioning level i.e at pass one. Pass two, the sorting ordering pass, is not with-out its problems. They are problems particular to Arabic and they loom large in more than one computer system. An example not the most serious, is noticed when the numerical code assigned to Arabic does not follow the order the alphabet follows. The outcome hampers writing easy search sort routines of standard algorithms.

A similar situation crops when the different graphical shapes of certain Arabic characters are represented e.g ك and ک ی and ی ect or when certain common motiffs like ں are coded and included for use in conjunction with other shapes like ہ or ہ to produce ہں or ہں when joined. Here the programmer will be faced with a problem of a different type i.e where in the alphabetic table will he fit these additions. Even when vowels are included in

schemes like ASMO or CODAR-U the same question will be asked as to where do the vowels rank in the conventional table of Arabic alphabet.

3. SOME SOLUTIONS

This paper attempts to solve partially and not totally a small part and not all parts of the numerous problems posed. It concentrates on pass one, the conditioning pass of the three-pass word-frequency table producting programme. Two different attempts are made. One is simple and unconcentional. The other will show more sophistication while retaining the conventions of Arabic transcription.

3.1. Scheme one

In this scheme the task of pass one is extremely eased by disintegrating composite words at text entry stage. This is done by the operator separating between the component words of a composite word by simple insertion of space characters. The composite Arabic word ليعطيهم will enter as ل يعطى هم under such a strategy the conventional punctuation marks can be used by the programme as separators between actual Arabic words without the need for any further processing. Such a scheme coud be blamed for its obvious departure from the traditions of Arabic transcription and for the strain and burden it taxes at the data entry stage.

3.2. Scheme Two

The second suggested conditioning programme demands more work from both machine and operator. Here the text is entered according to the rules of written Arabic. The machine will then serially output the text words as they are, composite or simple, on the system terminal in a word per line format and prompts the operator to classify each word as noun verb or preposition (N, V OR P). The class N, V or P is given according to the nature of the word if it is a simple word and according to the focal word of a composit word. This direct information will later be used by the machine in conjunction with a variety of permenant short tables of certain selected words in disintegrating composite words to a focal word and affixes.

The affixes to be separated from nouns might come at the beginings or ends of words, of those that come at the begining are ل , ب , ك و ال etc and of those that come at the end are the possessive pronouns like ه , ها , هم etc.

When separating prefixes to nouns or verbs the program depends on look-up table containing all Arabic words of the class identified and begining with the same charaters as the prefix under scrutiny. If the affix to be separated is a suffix then the table to be searched should contain all Arabic words of the class indentified ending with the same characters as the suffix to be isolated.

The lookup tables mentioned are rather short tables. The longest among them is the table of noun words ending with ن — it is 120 words long (1) on the other hand Arabic words ending with هم are approximately twenty. Most of the remaining tables tend to vary between twenty and forty indicating reasonable storage requirements and short lookup time.

The programme has three different routes corresponding to the three different classes of words nouns verb or prepositions. See Figure 1

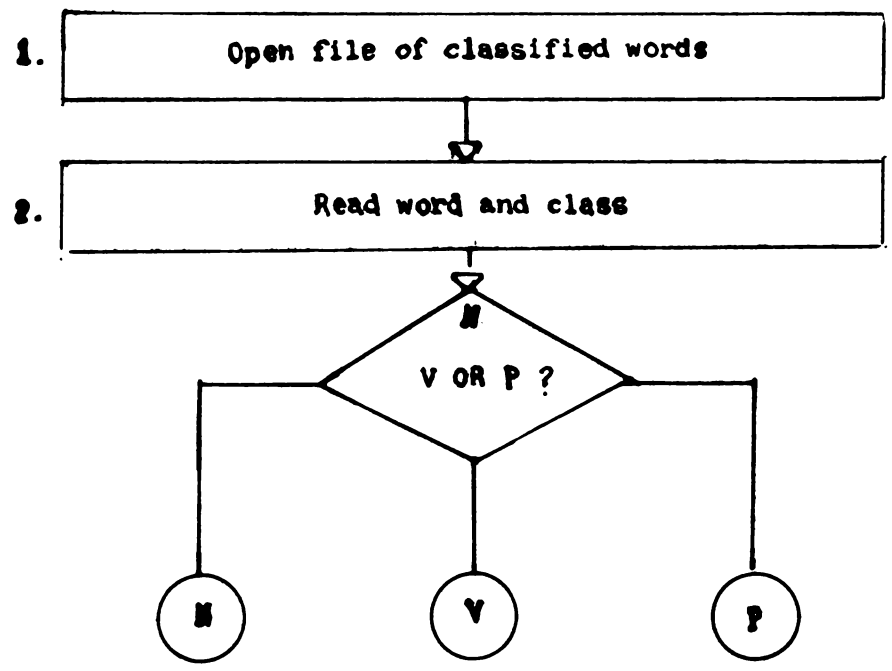


Figure 1.
Programme Routes

The noun's route-see Figure 2 - is designed to separate prefix only as well as suffixes ه , ها and the rest of the possessive pronouns. Other noun prefixes and suffixes could be incorporated if need be in the same route.

Verb affixes are separated in a manner similar to that applied to nouns. Of all the prefixes of verbs only س the futuristic indicator is sorted and isolated. The program searches a table

Figure 2.
Noun Processor

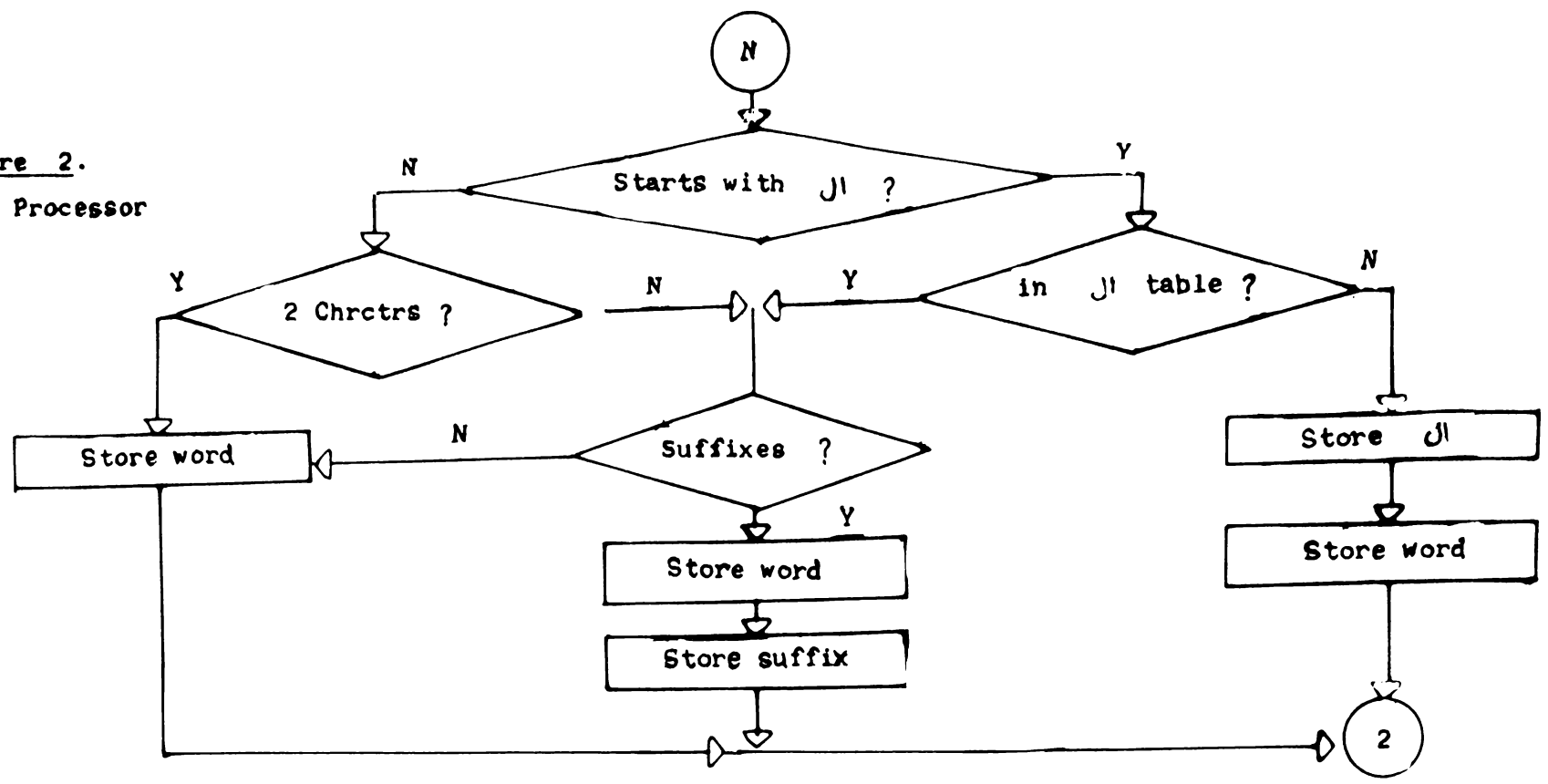


Figure 3.
Verb Processor

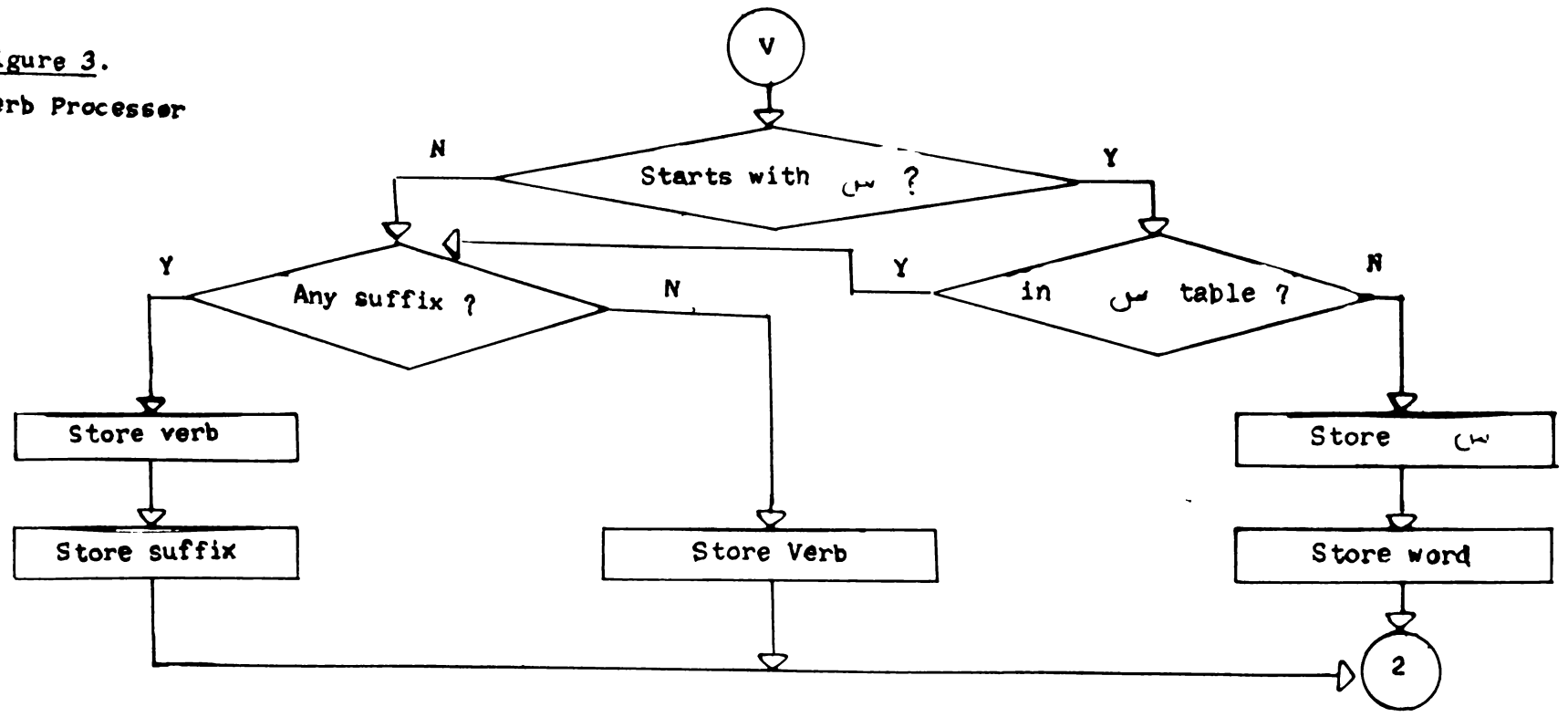
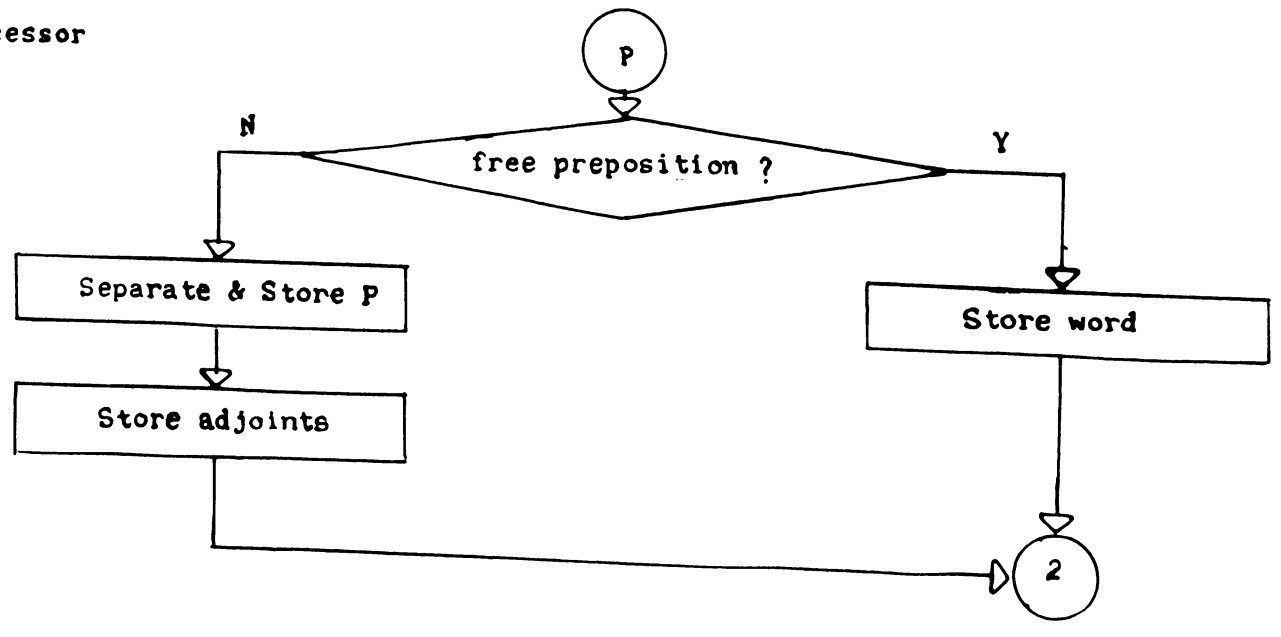


Figure 4.
Preposition Processor



of all verbs starting with س whenever a verb word with a س leading character is detected. Sucess of search means that س is an integral part of word . Failure means that the character is a prefix to be isolated, see Figure 3.

To separate suffixes from verbs, the verbs trailing character or characters are identified and then compared with suffixes like هم و ها و ه etc which when detected triggers a programme search i in the table of words ending with the characters detected. A successful search implies no suffix and failure suggests a suffix.

The strategies applied in processing prepositions are no different from those applied to nouns and verbs. Prepositions are written in Arabic texts either as free prepositions or as prefixes. when-ever a word designated as P is detected the programme compares it to the contents of the table of free prepositions. This search determines whether the preposition is free or a suffix, see Figure 4.

4. CONCLUSION

The programme is not expected to dissect all composite words to their ingredients. It is certain that nouns adjoined to some prepositions will filter through without detection. Verb prefixes like ف and ل are not catered for either. However the experimentation on a Limited Arabic text has shw show that nearly 85% of the composite words are processed and reduced to their component words. Further refinements and deeper sophistication might be added to tackle noun prefixes other than الـ and verb prefixes apart from س . A 90% mark of detection might then be arrained and possibly surppassed. With such success the author hopes that the scope of application of such techniques might be extended from generating frequency tables to the realm of machine translation and similar fields. One also hopes that fielding such problems might lure the linguists into joining forces with technolgists and others in an integrated effort to solve them.

REFERENCES

1. Hava, J. G., Arabic English Dictionary, Dar El Mashreg, Beirut, 1970.

BILINGUAL HARDWARE AND SOFTWARE: CASE STUDIES

Case Study on the Design of Bilingual Hardware and Software

PHYS. J. KARLSON
SIEMENS AG
München, FRG

1. Introduction

With the increasing number of mainframe installations in the Arab World, additional effort in Arabization of peripheral devices, user software and system components is required.

The first steps of arabization consisted of adaptations for existing peripheral devices for the Arab Countries. Usually, these devices had only a second character set, and a few features for controlling display mode, cursor movement and a few related functions. Code tables and character sets were defined by the manufacturers, and device handling was complicated both for the programmers and the terminal operators.

Within the last 5 years, the microprocessor and associated software has brought intelligence into terminals and printers, forming a new base for solving language-specific problems.

Standardization of character sets and codes is nowadays laid down in standards and de-facto standards.
The standard ASMO 449 and the proposal of ECMA/TC1 AWG are generally accepted by users and manufacturers.
Unfortunately it will take time, until these standards become important part of tenders and proposals.

Moreover, many details of realization are still up to the manufacturers of data terminal equipment and software engineers.

1.1 Customer requirements

Apart from the area of science and technology, large DP-systems are used in the public sector of administration, in industry and commerce. Most users still prefer to set up their own custom-tailored application packages.

Development aspects for bilingual systems have to take this situation into account.

Important features are:

- Usage of existing high-level-languages (PL/1, COBOL, FORTRAN) according to the standards for bilingual application programming.
- Existence of software tools for program development, data entry, and testing.
- Arabized system environment; no need for assembler programming.
- Code standardization for data exchange.
- User-friendly operation of both data terminal equipment and application packages.

Features which are still of minor importance are calligraphic quality and performance.

1.2 Characteristics of script and language

28 basic characters form the arabic alphabet. Arabic text is always written from right to left in cursive script.
Depending on the requirements of calligraphic quality, about 65 - 90 character shapes are required in an input/output device. Adding vowels and diacritics extends the character set by another 8 - 14 characters. If all aspects of linguistic research, country-specific peculiarities and calligraphy would be taken into consideration, a character set of approximately 600 characters would be necessary.

Because of the necessity to change the write direction, bilingual writing brings up problems in writing decimal numbers and arithmetic expressions.

1.3 Data Processing and its historical shortcomings

Both hardware and software components of todays data processing equipment still have built-in shortcomings, as there are

- a very limited number of codes usable for printable characters,
- a large number of codes unusable for printable characters, but also unused by the system

- limits such as the 80 character limit per line on data display terminals, the continuation line technique, and similar limits inherited from punched-card batch systems.

There are trends to overcome these restrictions. However, aspects of compatibility still have a higher priority for both customer and manufacturer.

Almost all existing high-level languages feature only statements providing record-oriented input and output operations on terminals Control characters inside strings or inside a group of statements resulting in one output operation are rarely supported by the language.

2. Representation of Arabic in operating systems

A modern mainframe timesharing operating system comprises between two and five million lines of code, usually written in assembly language. Main tasks of the operating systems are device handling, memory management, task management, spool handling, data management, and communication access support.

From the user's point of view, compilers and associated run-time systems, data base systems, and transaction monitors also belong to the system.

It is evident that the arabization of an existing operating system is a large effort. Arabization of compilers, query systems, and similar software is complicated and destroys the portability of existing application packages unless the existing standards are extended in a suitable way.
The balance of this paper discusses the efforts made to modify the BS2000/PDN Operating System to provide a suitable system interface for writing arabic application packages.

2.1 BS2000 Operating system characteristics

The Operating System BS2000 is designed for medium to large-scale data processing systems.

Its main features may be summarized as follows:
- Virtual memory concept
- Time slicing
- Fully integrated interactive capability
- Uniform command language
- Modular structure

All modes of operation (batch processing, interactive processing, and transaction processing) can be handled simultaneously.

Components of BS2000 which are involved in the Arabization process are shown schematically in Fig.1.

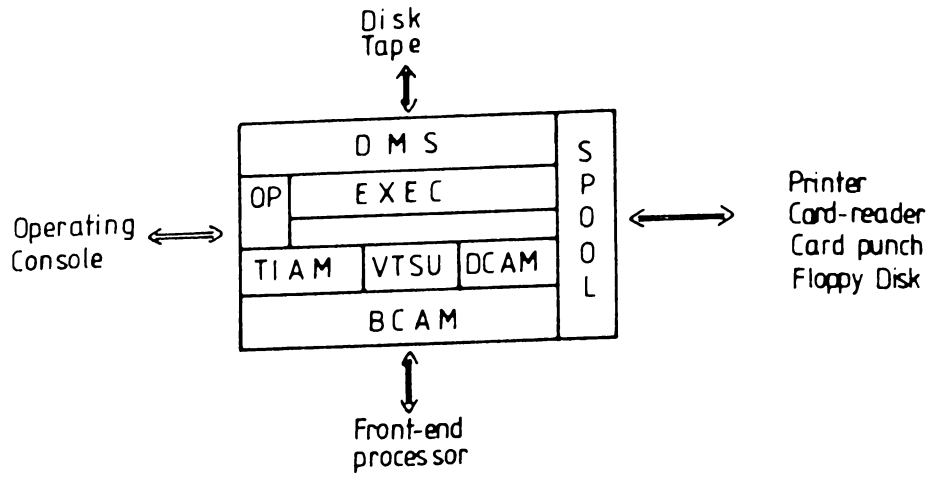


Fig. 1: Components involved in the Arabization

The VTSU (Virtual terminal support unit) is an interface between the basic communication access components (BCAM) and the user interfaces for interactive processing and transaction processing.

The SPOOL system handles the operation of peripheral devices, such as high-speed printers, card readers, card punch, printers, and floppy disk drives.

2.2 PDN Subsystem for data communication and networking

The Operating system PDN is designed for front-end processors and terminal computers. Its main task is concentration and distribution of messages exchanged between various stations and the host computer. A station may be a terminal or an application program running in the communication computer.

Arabic message handling can now be performed by defining a station represented by a privileged program. Fig. 2 shows the relation between data communication software in the host computer (BS2000 DCM) and the communication computer (PDN).

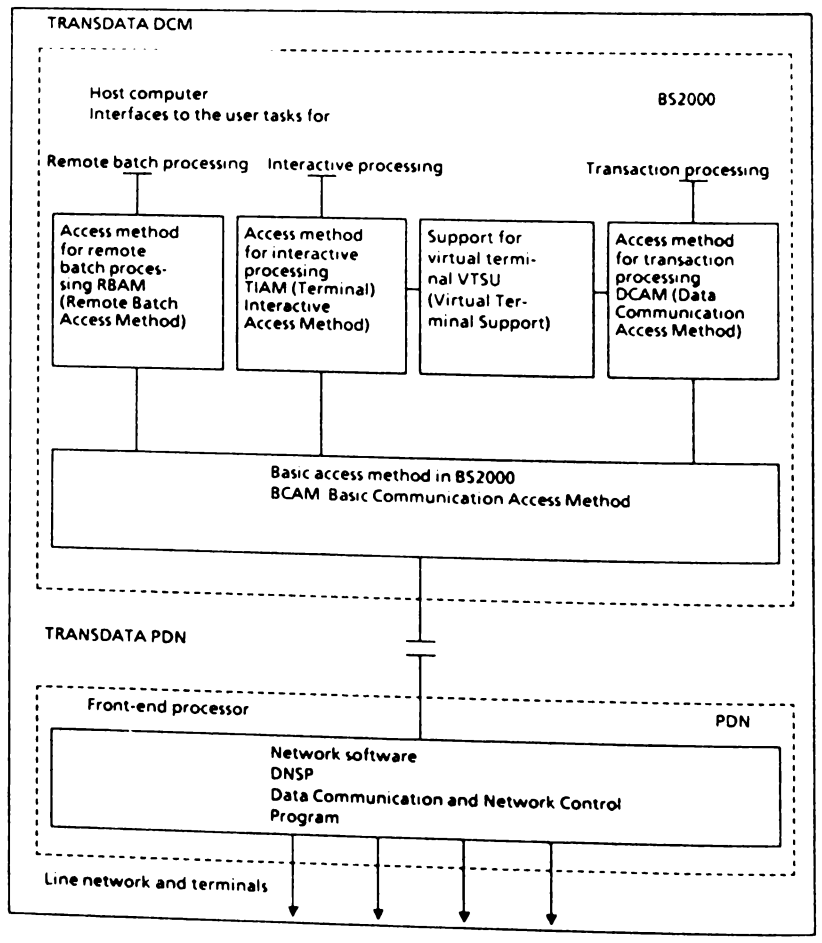


Fig. 2: Interfaces host-software and network software

2.3 Implementation of national alphabets

In BS2000, EBCDIC is used as host code. The terminals normally are driven with ASCII, and any other 7-bit code can be introduced by defining its code table inside PDN at network generation time.

Code conversion EBCDIC to ASCII (for a message from the host to a terminal) and ASCII to EBCDIC (for a terminal input to the system or to an application) is performed by PDN.

The implementation of national alphabets for BS2000 can be summarized in the following steps:

- Design of an extended EBCDIC code table
- Selection of the line code
- Definition of code conversion tables

These technical measures ensure that a supervisor call "write to terminal" using the extended coding can be executed.

The necessary modifications should be as transparent to the user as possible. Future development of software, hardware and progress in standardization may cause conflicts with components involved in code extension and code conversion.

3. Bilingual terminals and printers

Most of the data display terminals and the workstation printers available in the market today already have built-in microprocessors and associated firmware. Thus, implementation of the necessary functions for bilingual devices is not a big problem. From the manufacturer's point of view it is easier to implement functions in the firmware of a specific device than to change functions in an existing operating system.

3.1 Arabic character generator

The main problem for the design of an arabic character generator is the dot matrix format of the devices.
Arabic characters are wider than latin characters; a good representation could be obtained by turning the square of the dot matrix by 90^{o}. Unfortunately, this produces major conflicts with the hardware characteristics (spacing, number of lines, number of characters per line).
Introduction of variable width character cells only makes sense in conjunction with a sophisticated editing software.

3.2 Contextual analysis

Contextual analysis is preferably carried out by the firmware of data terminal equipment. The algorithm occupies about 300 - 500 instructions in assembly language, and the necessary tables for conversion of character shapes need about 1 K.

3.3 Terminal operation

Terminal operation should be as simple as possible. This demand is much harder to fulfil than it appears.

Important features for bilingual terminal operation can be summarized as follows:

- Automatic contextual analysis
- Arabic number operation
- Language-dependent cursor handling
- Screen mask operation with program-controlled output mode
- Field attributes "Arabic", "Latin".

Automatic contextual analysis must be enabled as long as the arabic character generator is enabled. Correction of misspelled words is normally done by overtyping the wrong letters by the correct ones. This produces a logical problem, because the correct shape may be dependent on subsequent corrections. The simplest method consists of bringing up a likely representation and analyzing the word once more after the cursor is moved outside the word to be corrected.

Arabic numerals are written from left to right. In other words, they are to be treated like latin insertions in an arabic text.

For users dealing only with word processing applications, a second mechanism should be integrated for keying in numerals in reverse order, as in normal typewriting.

Screen masks are a necessary aid for entering large amount of data efficiently. Bilingual masks can be, for instance, latin masks with arabic comments or arabic masks with additional latin fields. Mode switching is to be carried out by the application, not by the terminal operator.

Variable fields can be latin fields or arabic fields. Mixed latin and arabic fields do not make much sense, but their definition should be possible.
The field attributes "arabic" and "latin" should evoke the associated character generator automatically, without depressing any key on the keyboard.
A software-controlled lock can prevent unintentional switchover of mode and character generator.

3.4 Device handling support

A general approach towards arabization cannot stop at device level. It is not enough to have bilingual terminals and printers and a list of escape sequences which could be used by an experienced systems programmer for device control.

All functions mentioned above must be implemented into an existing operating system under the following conditions:

- all existing standard software shall run on bilingual devices in english without any modifications

- arabic functions must be accessible to standard software products in a very simple way
- necessary modifications in the operating system should not affect proper working of the system software or existing latin applications.

Dedicated device support for bilingual terminals and printers must be offered by the manufacturer of a DP-system, if the system is used by the customer for developing tailored application packages.

4. Phases of Arabization for BS2000/PDN

Arabization of BS2000/PDN started in 1980. Since 1982 larger efforts have been made to embed bilingual data terminal equipment into the operating system. The concept presented herein does not always make use of the latest standards, since aspects of compatibility with existing systems and devices have a higher priority for the customer.

4.1 Code Tables

As already pointed out, the SIEMENS series 7.500 computer uses EBCDIC as host code. Our first task was to implement the basic arabic character set into the existing EBCDIC table without affecting the existing coding. The present solution is shown in Fig. 3.

	4	5	6	7	8	9	A	B	C	D	E	F
0	SP	&	-	ف				٠		—	SP	0
1	ء	ت	/	ق	a	j		١	A	J		1
2	آ	ث	س	ك	b	k	s	٢	B	K	S	2
3	أ	ج	ش	ل	c	l	t	٣	C	L	T	3
4	ؤ	ح	ص	م	d	m	u	٤	D	M	U	4
5	إ	خ	ض	ن	e	n	v	٥	E	N	V	5
6	ئ	د	ط	ه	f	o	w	٦	F	O	W	6
7	ا	ذ	ظ	و	g	p	x	٧	G	P	X	7
8	ب	ر	ع	ى	h	q	y	٨	H	Q	Y	8
9	ة	ز	غ	ي	i	r	z	٩	I	R	Z	9
A	`	!	^	:	ﻷ	))	ﻷ		:	¨	!	
B	.	$	,	#	ـً	ـّ	'	[	.	¸	$	{
C	<	*	٪	ة	×	ـٌ	ـ	\	¸	٪	*	<
D	(	)	_	'	ـ	ـْ	ـ	]	¨	(	)	}
E	+	;	>	=	،	÷	؛	ﻵ	¸	>	=	+
F	|	—	?	"	((	،	؟	ﻵ	¨		"	¬

Fig. 3: Implementation of the arabic alphabet into EBCDIC

This table attempts to cover the following requirements:

- Compatibility with EBCDIC with regard to the latin character subset
- Full compatibility to ECMA/TC1 AWG
- Preparation of standard ASMO 449
- Full bilingual text processing
- Full vowelisation.

The arabic basic consonants are grouped in the range X'41' - X'49, X'51 - X'59', X'60', X'62' - X'69', X'70' - X'79' in accordance with the collating sequence of ASMO 449/82.

Vowels and ligatures are implemented in accordance with the ECMA proposal in order to maintain compatibility to existing application software.
Since sorting without preprocessing always produces classification errors, grouping of vowels is of minor importance.

Emphasis has been laid on a simple implementation of this table.

The front-end processor is driven by the host system in EBCDIC. The code conversion EBCDIC/ASCII, EBCDIC/ARCII or EBCDIC/ASMO is to be done in the front-end processor, thus offering the possibility to adapt the subsystem to the standard used by the customer, by simply exchanging 5 code conversion tables in PDN.
These tables are grouped in one module DTAB which can be altered by the end user if required.

4.2 The subsystem ARACOM-S V3.0

The subsystem for arabic communication on system level, ARACOM-S consists mainly of two components:

- the dual-language arabic/latin interface DALI. This module is part of the data communication and network supervisor program DNSP, the system of the front-end processor. It is the device driver for the bilingual terminals.

- the bilingual software for SPOOL-SE. This software comprises the privileged routines for contextual analysis on high-speed printer side, the associated analysis routines for the /PRINT command of the control system command language, and additional programs for communication with the computer operator and the system controller.

4.2.1 DALI

The main task of DALI is to insert the necessary escape sequences for the character generator into messages from the host to the terminals and to remove them from messages from terminal to the host.
The output mode of a message is determined either explicitely by the user, or implicitely by the first character of the message. This technique enables use of arabic strings like latin strings; there is no need to use escape sequences in the user program. Since most high-level languages do not permit control characters inside strings, and since all input/output operations are line-oriented, this is probably the best way to avoid complex extensions of existing compilers and their run-time systems.
The interface used for bilingual screen masks will be discussed in time.

4.2.2 SPOOL-SE

Bilingual printing is supported by SPOOL-SE. From the user's point of view, there is only a very very small difference in applying the /PRINT-command of the control system comand language.

By means of the FORM-parameter, the user defines the printer form (arabic or latin) and the types and numbers of printers if there is more than one type or if several devices are available.
For each printer, a set of code tables is held in privileged memory areas, since the translation basic character to printer font differs between the models.

4.3 The subsystem ARACOM-U V3.0

ARACOM-U is the user interface to ARACOM-S. Its components are

- the macro library ARACOM.MACLIB
  These macros offer functions for assembler programmers which are not available on high-level language basis. For instance, simple screen masks can be set up using these macros.

- the module library ARACOM.MODLIB
  A set of routines is offered which can be called in any high-level-language, such as COBOL and FORTRAN, in order to perform special I/O on bilingual terminals. For instance, the module READLAT reads an arabic string in latin mode from the terminal. Sort preprocessing and postprocessing routines will be available in the near future.

- Utilities
  For instance, a bilingual file print on a locally connected workstation printer can be produced using the utility HCPRINT.

4.4 Arabization of software tools

The adaptation of existing software tools for the arabic language may be either a very complex task or a comparatively simple one, depending on the internal structure of the standard product. From the manufacturer's point of view, those tools are to be first arabized which are necessary for the arab application programmer:

- the file editor, which is the most important tool for programmers
- the screen mask generator, which allows for setting up a convenient interface between application program and terminal operator
- interactive query systems for databases, since they present end user functions.

4.5 Arabization of system interfaces

Among those interfaces offering device support for bilingual data terminal equipment, a more general approach is necessary to handle national alphabets throughout an operating system. The detailed description of necessary modifications is out of the scope of the current paper. The most essential precondition is that transparency for all printable characters (i.e. the range X'40' - X'FF' for EBCDIC and X'20' - X'7F') for ASCII and all other 7-bit codes is guaranteed.
If restrictions are unavoidable, they have to occur in dedicated modules which can be altered by the customer.

4.6 Arabic interaction on system level

The previous paragraphs showed the efforts made for arabization of the programmer's interfaces. The question now arises, which measures could be taken to arabize man-machine interfaces:

- The operator task, responsible for the dialog between the system and the computer operator.
- The control system command language of the executive and the data management system.
- The system message file.
- Utility routines.

Generally, comparatively high costs will arise in introducing these features into the host system, and we doubt that an application programmer will really need an arabic control system command language.
The history of data processing proves that national development of operating systems and programming languages can be a waste of time and money.

Functions which are really helpful are an arabic message file, and a partially arabized control system command language.

5. The bilingual data display terminal DSS3975-A50

5.1 Product definition and specification

The bilingual terminal DSS3975 was defined at the end of 1983.
Its predecessor, the ZBE3974R went out of production, thus providing an opportunity to make bigger steps towards arabization.

The main design aspects for this device can be summarized as follows:

- Full compatibility with existing latin terminals.
- Full arabic character set including vowels.
- Shift-in operation in latin and arabic mode.
- Contextual analysis.
- Field attributes "arabic", "latin".
- Software-controlled character generator loadable by the host, and readable by the host.
- Symbol construction program.
- Logical data transmission.

The following section discusses these important features.

5.1.1 Logical data transmission

In ordinary data display terminals, linear scanning of the screen image (its counterpart in a RAM area) is performed. Consequently, arabic strings as an insertion in a latin line (for instance a source line of a program) would be transmitted to the computer in reverse order. The same applies to latin insertion in an arabic line.

This problem can be overcome by introducing a logical scanning of the screen content for arabic mode and latin mode, as shown schematically in Fig. 4.

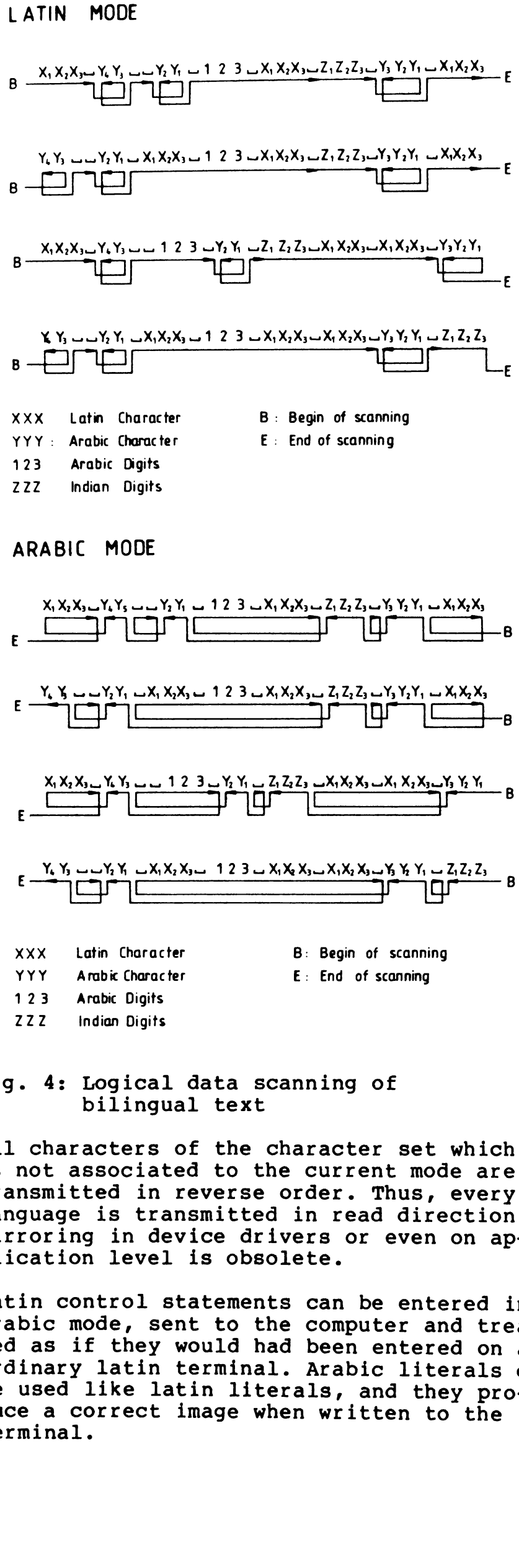


Fig. 4: Logical data scanning of bilingual text

All characters of the character set which is not associated to the current mode are transmitted in reverse order. Thus, every language is transmitted in read direction. Mirroring in device drivers or even on application level is obsolete.

Latin control statements can be entered in arabic mode, sent to the computer and treated as if they would had been entered on an ordinary latin terminal. Arabic literals can be used like latin literals, and they produce a correct image when written to the terminal.

5.1.2 Screen mask operation

With screen formats, switchover of the character generator by the terminal operator becomes superfluous if variable fields can be defined as latin fields and arabic fields, bringing up automatically the associated character generator and operation mode (normal or shift-in) for this field. This feature is available in both latin and arabic mode.

5.1.3 Software-controlled character generator, symbol construction program

The character generator is part of the firmware. However, this is only the default setting. During start-up phase of the terminal, the character generator is copied into a NVRAM-area. From this area, the actual character generator is taken. It can be transmitted to the host on request by the host, or a customer-specific character generator can overwrite the standard character generator.

The complete character generator is 2611 bytes; it is transmitted in printable format.

The symbol construction program enables editing of the character generator, to extend it by a maximum of 16 additional characters, to send the edited character generator to a EPROM-programmer, or to print it on a locally connected workstation printer in matrix format (Fig.5).

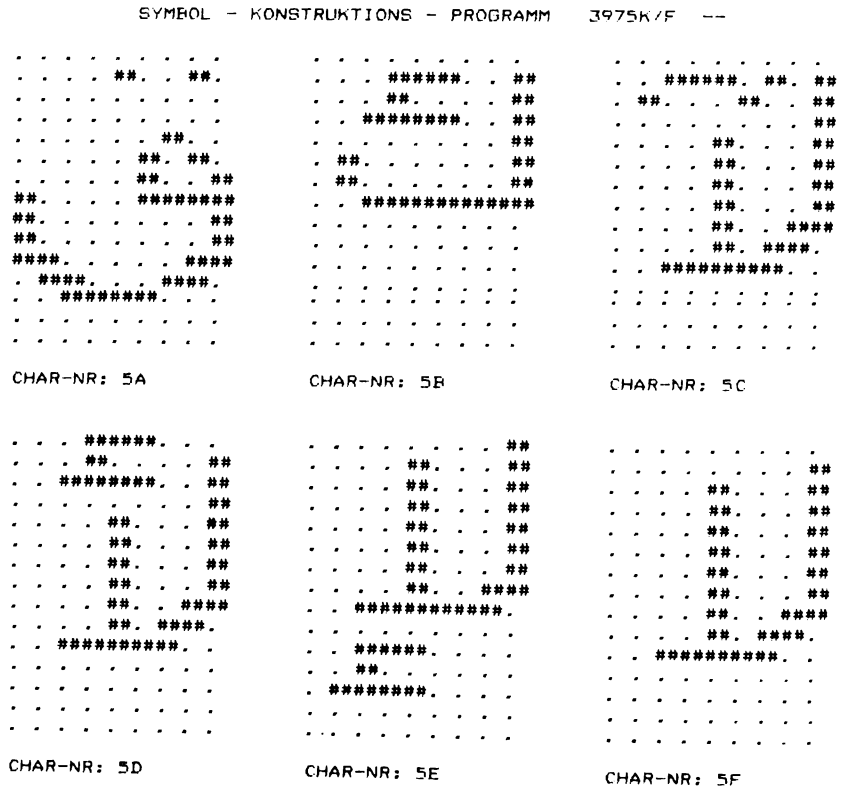


Fig. 5: Printout of some standard symbols in matrix form

5.2 Interfaces

5.2.1 Hardware

The DSS3975A50 has a V24-Interface for communication with a front-end processor or terminal computer.
A V24-interface for a local work station printer is provided.
The terminal is operated with MSV1-protocol, and the printer is driven without link protocol.

5.2.2 Software

Description of the software interface in detail is out of the scope of the current paper.
Nevertheless, some details shall be presented here, since this technique is applicable in similar cases. Each message, exchanged between terminal and computer, is prefixed by one or several message headers.

These headers control terminal operation, the subset of function keys available for the terminal operator, operation of local workstations, and other functions.

The parameter area PARØØL of the message header and the bit setting used for the DSS3975-A50 are shown in Fig. 6.

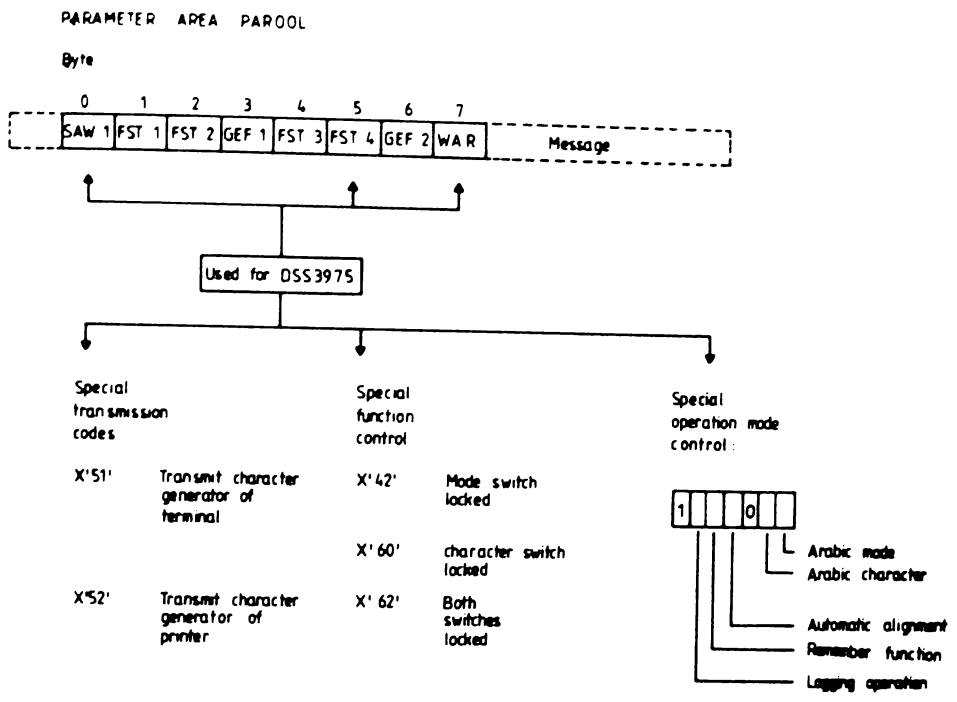


Fig. 6: Message header interface of the DSS3975

Standard software products like editor, screen mask generator, etc., define this area, and the setting of bits of the selected fields is irrelevant for the type of terminal we use.

The advantages are obvious:

- For mode switching no insertion of ESC-sequences into output messages is necessary. Message length and message structure are unaltered.

- Additional information for the operation of the bilingual terminal can be coded, for instance, for automatic alignment of non-mode conform data fields.

5.3 Realization

Although the terminal has many precious features, development of the microprocessor software by a highly-experienced team of 3 engineers took about 5 months. The software is stored in 80 k EPROM; 36 k RAM working storage are needed.

Fig. 7 shows a block diagram of the terminal firmware.

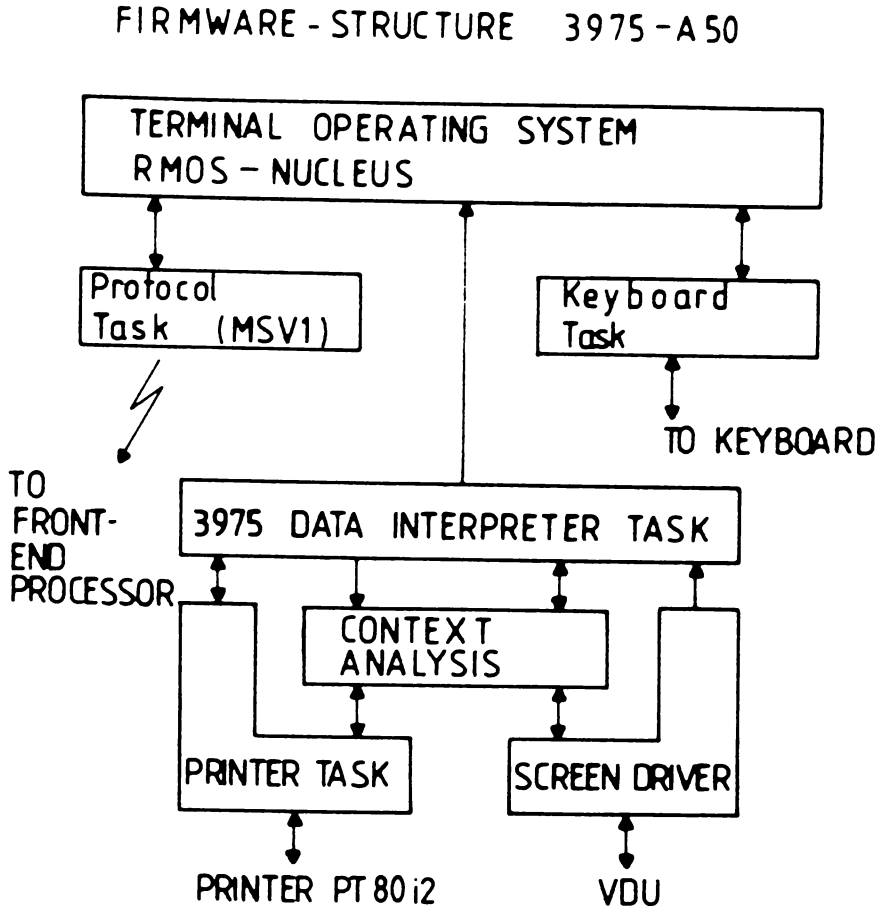


Fig. 7: Block diagram of the terminal firmware structure

6. Bilingual printer 3338/II

The 3338 is a high-speed band printer with a printing speed of 1250 lines per minute. For all printers employing mechanical parts like drum, belt, or type train, the problem arises that for bilingual printing a comparatively large character set is required. Excellent quality of calligraphic shape is only possible without "optimization", i.e. each arabic character shape has to be on the band. Decomposition techniques find their limitation in mechanical tolerances of the type face and in the phasing. Hammer activation is usually done by unloading a capacitor via a magnetic coil. Tolerances in capacity, inductive reactance, and associated amplifiers and thyristors produce smear effects or slight misalignment.
The decomposition of basic shapes is therefore not very promising. For example a ل (Lam final) cannot be composed by a vertical bar and an end tail like in س .
However, composition of independent elements can be a solution, even if printing speed goes down.

Fig. 8 shows the font selected for the 3338 band printer. Overprinting is used for dots, hamza and vowels.

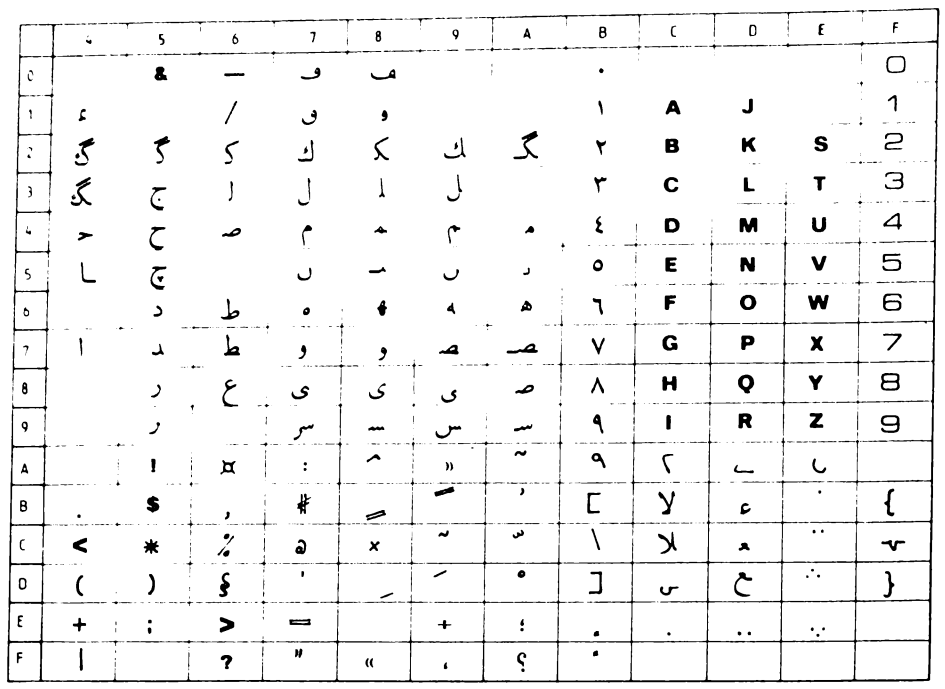


Fig. 8: Printer font of band printer 3338/II

The print band has 168 characters. Some character shapes are not essential. We support 2 different shapes of ٩ , 2 different shapes of "Heh middle". Customer-tailored print bands can be produced with a reduced font set, thus increasing printing speed.

There are 448 positions on the band. Fig. 9 shows a relative frequency analysis for the basic arabic characters. The number of repetitions is chosen according to the expected frequency of each letter; it varies between 1 and 5 for the arabic consonants.

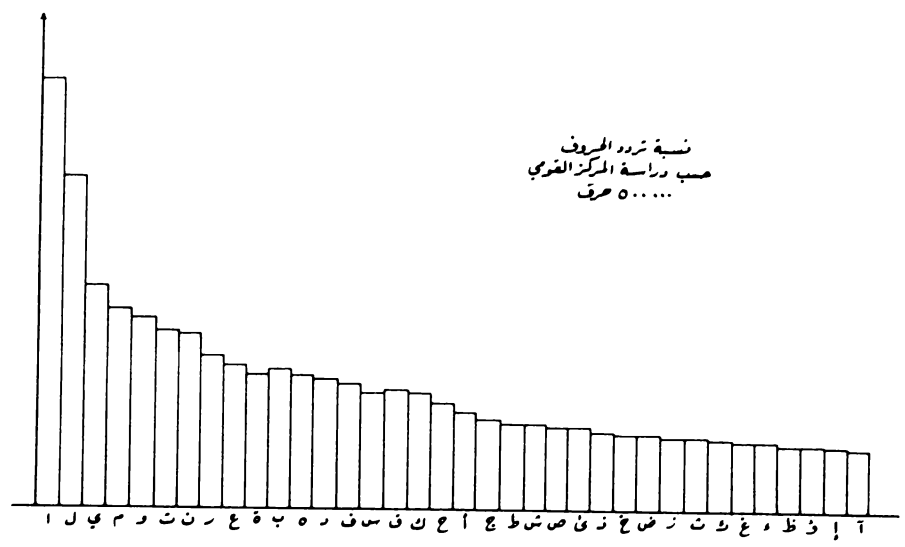


Fig. 9: Relative likelihood of arabic consonants

7. Ink-jet printer PT80i2, LED-printer 9025

The PT80i2 is a dot-matrix ink-jet printer with a character matrix of 13 x 12. Since the standard device is equipped with a loadable character generator, arabization simply consists of developing an arabic font for it. Contextual analysis is done in the DSS3975 if the printer is used as a work station printer. As a remote printer, the PT80i2 is connected directly to the front-end processor.

In this case, contextual analysis is carried out in the software GEPARD, a product supporting various types of remote printers.

Arabization of the LED-printer 9025 will bring about a perfect solution for the calligraphic problems. This page printer with a resolution of 400 dots per mm is equipped with a character generator of 256 x 256 characters stored on a Winchester disk. Laser scanner and electronic editing techniques allow for reading the character set directly from drawings of the single elements. The printer supports proportional script, and vowels are placed on the associated consonants. Unfortunately, development of the editing software is a very complex task.

8. Tools and Utilities

The features of the bilingual terminal and the system support offered by DALI give an excellent environment for arabization of existing tools and utilities.

It is difficult to inaugurate change requests for existing software packages; the technique presented here for ARAB-FORMPLAG and ARAB-EDTV16 does not require change requests. The only prerequisite is that the product is available in a library (module by module). A dedicated interface for bilingual terminals is then developed and added to the library.

The design of the bilingual terminal discussed previously, and the device driver DALI allow for

- Output of latin messages in arabic mode: No mirroring is necessary
- Input of latin messages in arabic mode
- Correct, mode-independent representation of arabic strings in logical order; The first byte in storage is the first letter of the arabic word
- Definition of arabic fields
- Terminal control without escape sequences.

Some details of the arabization of FORMPLAG and EDTV16 highlight the fact that small add-on modules can transform standard tools into arabized tools.

8.1 The bilingual screen mask generator ARAB-FORMPLAG

The requirement for a simple and general programming aid to support the use of masks displayed on a VDU terminal has been met by FORMPLAG, a mask generator incorporating type-checking facilities. FORMPLAG generates the relevant program sections as assembler modules.

Mask implementation is done via an interface module FORMBI. The user program adresses the desired action to the interface module in a subprogram call.
3 parameters are required: Control area, input-output area, and data area.

Arabization of this software tool must allow the user to control:
- the screen mask output mode
- the definition of variables as latin (standard) or arabic variables.

This information was easy to encode, since at mask generation time, a parameter ID = 'string' can be specified, which is laid down in the mask module.

Evaluation of this information is done at run-time. If the user prepares a screen mask for terminal output, the necessary modifications in the message header and the field-control-characters are carried out.

Alphabetic check of arabic strings entered into variables is carried out correctly if the internal check table of the product is modified.

Limit and range checks on numeric items are automatically supported, since the data display terminal transmits numerals in arabic mode from left to right. Usage of indian numbers currently assumes a code translation by the application.

8.2 The bilingual file editor ARAB-EDTV16

Arabization of a file editor is quite different from the function scope of a word processing system.
Word processing has only to ensure storage and retrieval of text in any internal data format; the file editor cannot make use of any control characters which would become part of the file after writing or rewriting it.

For a file editor, the scope of application lies especially in creation, update and maintenance of source programs and data files.

In case of the EDTV16, the terminal interface consists of calls for the FORMPLAG interface FORMBI (see above). Manipulation of files can be done by 1-letter control statements written in the first column of the respective lines, or by complex statements, written in a command line.

For instance, the statement
ON 1-12: 44-60F 'البيت','IP','الحديقة و'
inserts the word 'الحديقة و' before 'البيت' if 'البيت' is found in lines 1-12 between columns 44-60.

The design principles of the terminal and the device driver DALI, permit the original screen masks to be used without any modification for the arabic mode. The text field is defined as arabic field, and the first column is defined as latin field for the 1-letter control statements.

Switchover of latin mode to arabic mode is controlled via 2 function keys.

The main task consisted of analyzing the complex control statements. After development of the necessary routines, it turned out that the terminal handling becomes very complicated. Therefore windowing technique has been applied to facilitate entry of the ON-statement and certain similar statements.

9. Future aspects of Arabization

9.1 Device features

The cost of hardware components like microprocessors, memory chips, etc., is still minor in respect to other costs. Dot matrix size for the next bilingual terminal generation will be 16 x 24; for bilingual matrix printers, existing devices offer a very good resolution. Extensions of the firmware could bring up indexed character generation and automatic editing functions for correct vowelized arabic proportional script, without losing transparency.

Down-loadable character generators and associated editors will allow adaptations of the character set to customer demands without influencing system software, application software, or coding.

9.2 Trends of standardization

The standardization process for the arabic keyboard layout will hopefully be terminated within the next few months.
Standardization of contextual analysis algorithm, text editing algorithms for full or partially vowelized arabic would be helpful for the manufacturers.
Many investments for bilingual data terminal equipment are blocked by the lack of standards, because redesign of existing devices will be unavoidable, once the Arab countries agree upon a standard.
Portability of application packages can be achieved by developing strict recommendations for the implementation of mode switching, character generator selection, and the associated control characters for triggering these functions.

Recommendations on font design on at least a general-purpose level and a high quality calligraphic level would be very helpful.

Investments made by the manufacturers should have an influence on the competition, in that non-standardized bilingual devices can no longer be offered to authorities, universities, and projects solicitated by the government. Such a policy could speed up the development of bilingual hardware and software in accordance with the requirements of the arab end users.

Acknowledgements

The author would like to thank Mr.H.Pohl and Dr.Drexler for supporting this case study, and to Mrs.Lange and Mrs.Remsing for typing the manuscript and drafting the figures.

METAL: An Operative Machine Translation System

THOMAS SCHNEIDER
SIEMENS AG
München, FRG

1. GOALS

It is a truism that the amount of technical documentation generated is growing. Innovation cycles are getting ever shorter, and this implies the necessity to provide documentation for new devices or processes more frequently than in the past. Moreover, the nature of the technology involved is changing. The mechanical implements of the nineteenth century were largely self-explanatory, being concrete and tangible, so that the documentation needed to operate the device could be kept to a minimum. Especially with the advent of miniaturization of devices and a gradual shift towards abstract implementations of problem-solving procedures like software, the user is no longer in a position to comprehend the workings of such a sophisticated system without detailed documentation.

The exponential growth of documentation, if expensive in itself, becomes even more of a problem when seen in the context of multilinguality. Economic interdependence of countries, and especially the need for technology transfer, requires that - along with the physical devices - relevant documentation is provided in a foreign language, in a language the user can fully understand.

In other words, not only the amount of documentation itself is growing but a larger percentage of it has to be translated. And since there are not enough qualified translators available, at least not for highly specialized technical fields, there is a definite need for high quality machine translation as a tool.

After some less than successful attempts at employing operative machine translation systems offered by commercial vendors, Siemens in 1978 entered into a cooperation with the University of Texas at Austin to develop the METAL system. This language-independent system is at present used successfully in a pilot application for technical translations from German into English. Other language pairs are currently under development or are planned for the medium-range future.

2. STRUCTURE

From the outset, METAL was built in a highly modular way so as to permit the inclusion of new elements or the modification of existing elements without any ill effect on the other components. This "open" structure also makes the system an adequate basis for future applications in semantic content analysis, information retrieval or as a natural language front-end for expert systems or data bases. Its first application, however, is machine translation.

As there is at present no linguistic theory available that would describe even a single language unambiguously and completely, a more or less eclectic approach has to be chosen.

2.1. Grammar

METAL uses basically phrase-structure rules which are augmented by tests on the constituents and various other constraints. As the rules are recursively applied, in a Markovian manner, their number can be kept low. To illustrate the advantages of a recursive system let us take the following sample rules:

1. S - NP VP
2. NP - DET ADJ N
3. ADJ - ADJ ADJ
4. ADJ - ADV ADJ

Rule 1 says that a sentence may consist of a noun phrase and a verb phrase, rule 2 that a noun phrase may consist of a determiner, an adjective, and a noun. Rules 3 and 4, however, state that an adjective may consist of two adjectives or of an adverb and an adjective, respectively (of course, all constraints and tests have been left out in our sample rules).

Now take the following sentences:

a. The old man works.

Two rules, 1 and 2, would interpret the surface structure as a sentence.

b. The very old man works.

Here, rules 1, 2, and 4 would lead to a sentence S.

c. The grey-haired old man works.

Rules 1,2, and 3 interpret the structure to be an S (according to rule 3, the two adjectives "old" and "grey-haired" are interpreted as one adjective for analysis in rule 2). Multiple application of rules 3 and 4, however, will interpret an (admittedly contrived) structure like:

d. The very sick grey-haired mildly insane friendly old man works.

A conventional MT system might try to account for every possible surface structure with a separate rule. Aside from the fact that for free-word-order languages this is simply impossible, managing tens of thousands of separate rules is quite a task. METAL at present uses not more than 600 rules but nevertheless is able to deal structurally with sentences it has never encountered before.

The grammar rules are indexed to make processing more efficient and to open the possibility of a partial use of the grammar for e.g. "quick-and-dirty" information gathering. The most commonly applied rules, e.g. for morphology and for frequently occurring basic structures, are defined as the most basic level. If a structure can be interpreted using lower-level rules then the more complex and less likely rules are disregarded, which saves processing time. So if for the purpose of a rough translation only the lower three levels of rules are invoked the translation result will be less than perfect but perhaps still quite sufficient for some applications, and it will be faster.

METAL uses a parallel bottom-up parser with some top-down filtering which prevents the possible exponential explosion of interpretations by eliminating unlikely paths via preferential weightings calculated from lexical and grammatical data. Thus it usually provides the best interpretation for transfer to the target language. If no plausible interpretation of the sentence is reached the system will go into a fail-soft routine and output the individual phrases it has been able to interpret. Surprisingly often, the output is still a grammatically correct translation of the original sentence (at least going from German into English); in other cases, the posteditor is called upon to correct the output.

Besides a section of tests on properties of constituents and their interaction, the analysis rules contain associated rule body procedures, i.e. information about which data is passed on in which form up the analysis tree to the higher nodes, and how structurally the phrase is to be dealt with in the target language.

METAL employs a transfer system rather than a meta-language. It was felt that to define a meta-language incorporating all possible features of many languages would not only be an endless task but the system would also collapse under its own weight. If, however, the meta-language were to be reduced to a manageable level of abstraction then too much information necessary for the translation process would be lost. And at least among Indo-European languages it was found that a transfer system was quite adequate.

2.2. Lexicon

The overall number of entries in a system dictionary is not a relevant criterion for the qualitative assessment of a machine translation system, or for a legitimate comparison of different systems. For one thing, the internal structure of an entry may differ. Perhaps all stems, or even all tokens of a word are listed separately in the dictionary, or, by contrast, all forms may be subsumed under a single entry, with internal pointers to various stems and word forms. METAL employs the latter structure.

Secondly, it makes a difference whether a system relies on one unidirectional dictionary, with a direct link between one source language word and one target language word, or whether multiple dictionaries are used.

METAL operates on both monolingual lexicons and a transfer lexicon. The monolingual dictionaries contain morphological, syntactic and semantic information needed for the analysis and generation of a language. The transfer lexicon provides the link from the source to the target language, indicating under which conditions, in which contextual environment and in which subject field the source language entry should point to which target language entry.

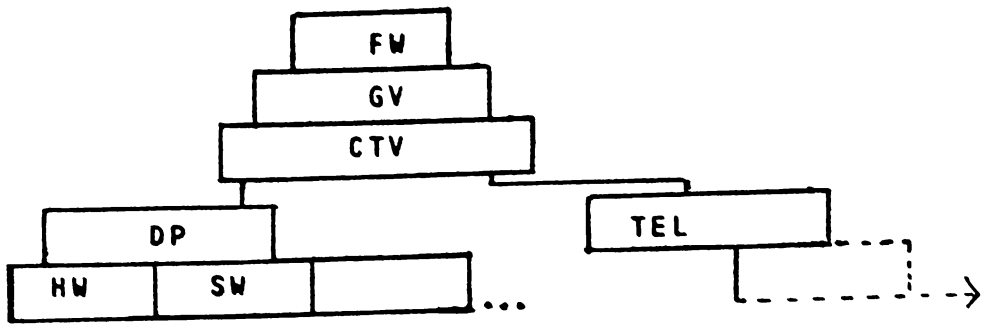
The advantages of such a structure, as used in METAL, are obvious. The extensive information contained in the monolingual dictionaries needs to be carried only once, even if many different entries in one of the languages correspond to the same entry in the other language. The transfers of the English verb "take", for example, may fill several pages of a book, and if each transfer entry were to contain the morphological and syntactic information of "take" as well, the system dictionary would be inflated excessively. This in turn would use much more storage space than necessary and make the system inefficient.

Another aspect of a system to be considered is the organization of its content. In most European languages, a set of 3000 to 4000 words make up approximately 85% of all texts (on the average). Beyond this limited set, the point of diminishing returns is soon reached. Increasing an undifferentiated general lexicon to more than 100 000 words, for example, would take a lot of storage space without decisively increasing text coverage. On the contrary, many unpleasant ambiguities would be introduced which can be avoided by a modular structure.

The METAL lexicon is organized as follows: There are modules for function words (FW) like articles, prepositions and conjunctions, for general vocabulary (GV) and for common technical vocabulary (CTV), organized in a tiered hierarchy. On the next level down, there are subject-specific modules which in turn can have subsets of more specific modules. The module "Data Processing", for example, has subsets called "Hardware", "Software", "Data Transmission" etc. Should any further specification be required for e.g. company-specific or device-specific terminology, new modules can be added to the structure.

The METAL dictionary structure is shown in figure 1 (simplified):

Figure 1 Dictionary Structure



Before a translation run is started, the modules appropriate to the text at hand are invoked. During the translation run, in dictionary lookup, the most specific modules are searched first. If no match is found there, the less specific modules are searched successively. This kind of search may be slower than a search procedure in which the modules of high frequency words are checked first, but it assures that the specific transfer needed for the subject area of the text receives the highest priority. In this way, METAL can be used for applications in different subject fields without structural modification.

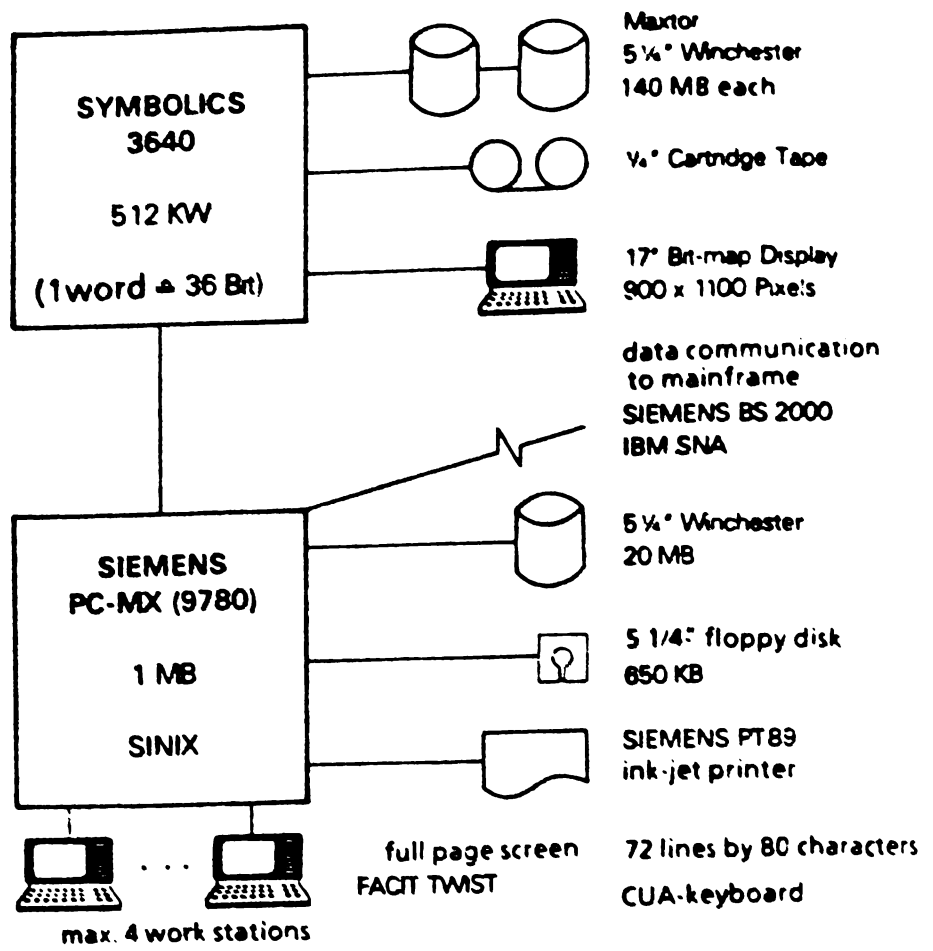
The main source for the required terminology for new subject fields is TEAM, the multilingual terminology data bank operated by Siemens. TEAM at present contains about 2 million records in many subject fields giving terms (plus supplementary information like definitions etc) in up to eight languages. An extension to include Arabic is in progress. As TEAM is used not just by Siemens but by many industrial partners and public institutions as well, it is assured that subject fields not covered by Siemens' internal spectrum are kept terminologically updated too.

An interface between TEAM and METAL facilitates the installation of new dictionary modules and the coding of new terms. A program called the DEFAULTER uses a set of rules as well as partial information already contained in the system dictionary to "guess" at morphological and syntactic behavior of new terms. In many cases the results only need to be proofread by the person responsible for the coding of the lexicon, and tests have indicated that the program reduces the time needed to code new terms by a factor of 10.

2.3. Hardware

At present, METAL is implemented in ZETALISP and runs on a Symbolics 3670 or 3640 LISP-machine. Since these LISP-machines are single-user stations but the system's throughput far exceeds the amount of text a single posteditor could handle, the LISP-machine has been linked to a multi-user PC (Siemens PC-MX). Thus METAL can run in batch in the background while formatting, post-editing and administration of the translation tasks are handled on the PC. The multi-user PC has interfaces to the BS2000 operating system as well as to UNIX/SINIX. The configuration looks as in Figure 2.

Figure 2 Hardware Configuration



To produce an acceptable machine translation system requires more than to build a prototype that will translate sentences. Most of the texts that need to be translated very quickly and are of great volume, i.e. those texts for which MT systems are intended, are heavily formatted. In some texts, more than half of the elements may be non-translatable material, e.g. flow charts, diagrams, tables and control characters for the generation of versatile print. It would be excessively expensive to manually extract the text portions to be translated and, after translation, manually reinput them. Besides that, it would create another source of errors.

Therefore, from the beginning, the MT system METAL was designed as part of a chain of processes, from text acquisition via deformatting, spelling correction, dictionary update and translation to specific postediting functions and reformatting procedures. The whole package is integrated in a translator work station which is marketed under the trade name of LITRAS.

3. STEPS OF A TRANSLATION RUN

A text is received in machine-readable form; it was either input originally at an office work station or was read in via OCR. The problem of having to handle non-standard, not machine-readable text is fortunately diminishing since most original texts nowadays are produced on a machine-readable medium to facilitate print and update of the document - whether intended for translation or not.

Several programs running on the PC-MX check the text pages for tables, graphs etc and mark them; they identify the text to be

translated and generate a mask of the page. The individual translation units, usually sentences but also single word headlines etc, are automatically recognized, numbered consecutively and extracted from the page mask. They are then passed on to the LISP-machine as an input file to METAL.

Before the actual translation takes place, the vocabulary of the text is compared with the system dictionary. It is advisable to update the dictionary before an extensive translation task to reduce the postediting that would result from having to correct unparsed sentences - unparsed because of missing words. Dictionary update is done either interactively at the terminal (menu-driven) or in batch (e.g. for bulk entries from TEAM). For shorter texts with few missing words the updating of the dictionary can be ignored, especially if the missing words are not likely to occur very often in future tasks. The resulting translation errors can then be corrected in postediting.

After translation, the individual translation units are automatically put into the same slots in the mask where the source language equivalent had been. The posteditor has the choice to either have source and target text appear in two windows on the same screen, or be shown just the reformatted target text, or look at an output of individual translation units in an interlinear source/target combination. In the latter case, he would start the reformatting procedures after postediting. The machine-readable postedited and automatically reformatted text with all the original control characters in place can then be transferred to printing systems or be processed further.

4. QUALITY

So far METAL has been used for the translation of more than a thousand pages of different texts from the subject fields of data processing and telecommunications. Translation speed at present is about 1 word per second which amounts to more than 200 pages per day.

Tests have shown that the quality of the raw machine output is superior to that of other commercial systems. However, it is very difficult to operate on the basis of a percentage of "correctness" since that may depend entirely on the type of text as well as on other factors. It is also quite difficult to decide if a "correctly" translated sentence ought to be polished stylistically or not. The tests showed that usually more than 60% of all sentences were translated "correctly", and that the rest was at least good enough to serve as a basis for a corrected version formulated by the post-editor.

More important, however, is the fact that translators (not just researchers) found the output acceptable enough to volunteer to use the system in their daily work. Spot checks showed an increase in productivity by a factor of 8, but this is an unrealistic goal considering the degree of concentration it takes to postedit large quantities of text and check for errors of content, perhaps produced in the original. Nevertheless, based on extensive pilot applications a productivity increase by a factor of 3 does seem realistic.

5. FUTURE APPLICATIONS

At present, METAL, integrated into the LITRAS translator work station, operates from German to English. Work on English as a source language and Spanish as a target language is under way. There are also firm plans for French and Dutch components.

Even though the development of a new language pair is no trivial matter, the existence of a robust, industrial-grade language-independent system with an extensive set of utilities will greatly reduce development time. It is estimated that the development of a new target language for German or English should require somewhere between eight and twelve man-years for an operable if limited system. Studies have shown that the present structure of METAL lends itself to an application of Arabic as a target language. For this, however, the cooperation of partners in the Arab World is required.

It should not be forgotten that the application of an advanced language analysis system like METAL does not stop at machine translation. Components of the system will in the future also be used for multilingual natural-language front-ends to expert systems, data bases and teaching and learning programs, not to mention goals like automatic abstracting and content-based information retrieval, e.g. for office or managerial systems.

Limitations in linguistic theory prevent the achievement of perfection but it seems that even the imperfect state-of-the-art as exemplified by METAL can contribute something to the solution of the problems imposed on us by multilinguality and by the flood of information.

Index

Affixes:
  affix separator, 112
  dictionary for, 203
  morphological analysis and, 201, 203, 209–217
  word frequency and, 218–221
Alif character, 11, 75, 112
Alkateb, 164
Allophones, 191
Alphabet (*see* Arabic script; Character repertoire)
Alphabetization, 111, 218
Al-Raed system, 56–57, 59
Al-Tariq, 164
American Standard Code for Information Interchange (ASCII), 107–108, 153
Apple II system, 56–57
Arabe Standard Voyelle (ASV), 108
Arabic language:
  intonation of, 132–136
  phonetics of, 130–132, 143–144
  statistical analysis of, 190–200
Arabic script:
  calligraphic definitions, 93–96
  character frequencies, 88–90, 97–101
  character recognition, 113–118
  context-sensitivity of, 73–74
  contours of, 173–175
  digitization of, 70–77
  layered character set, 110–112
  minimum alphabet for, 109
  morphological analysis of, 201–207, 208–217
  non-Arabic languages and, 107, 112
  particularities of, 72, 78–79, 114
  phonetic transcription of, 130–132
  standardization of (*see* Standardization)
  styles of, 80, 171
  (*See also* Character repertoire; *specific character names*)
Arabization (*see specific problems, systems*)
Arabrite, 164
Arab Standardization Metrology Organization (ASMO), 55, 102, 107–112, 153, 155
Arabstar, 164
ARACOM-S system 228–229
ARCII code, 153, 155
ARGOS system, 22
Articulatory synthesis, 127
Artificial intelligence, 19–28
  (*See also specific applications, systems*)
ASCII (*see* American Standard Code for Information Interchange)
ASMO (*see* Arab Standardizations Metrology Organization)
Assembly language, 45
Autoregressive modeling, 126

Ba character, 75
Baghdad NCC code, 108
Band printers, 42, 90–94
Bandwidth, 38
BASIC, 67, 153–160
Beam-search method, 26
Bilingual systems:
  character repertoire for, 152
  character standardization, 102–106
  customer requirements, 225
  design case study, 225–233
  display terminal for, 229
  evaluation of, 57–56
  hardware for, 56–57
  Indian languages and, 8–9
  keyboard design and, 97
  printing systems for, 102–103, 231–232
  software for, 231
  survey of, 55–68
  terminals for, 57
  text formatting and, 181–182
Blackboard model, 23–25, 44
Bottom-up control system, 25
Bunyan system, 161, 162

Calligraphy:
  calligraphic model, 72–77
  character definitions, 93–96
  typeface design and, 72–77
Cathode ray tube (CRT) display, 38–39, 69
Cepstrum, 120–121
Channel vocoder, 124, 125
Character generation (*see specific systems*)
Character matrix, 63, 79
Character recognition, 113–118, 188
Character repertoire, 79–80, 91–92, 103, 152–153, 228
  calligraphic definitions, 93–96
  layered set, 110–112
  sufficiency of, 107
Cluster analysis, 141
Coarticulation effect, 143
CODAR code, 108–109
Communications systems, 169–170, 228–229
  arabization of, 226
  office systems and, 29, 31
  software for, 161, 163
  work stations and, 35–36
Compilers, 45, 226
Composite characters, 7
Comprehensive morphological generator, 208–217
Computer(s):
  evaluation of, 59–61
  linguistics and, 185–189
  (*See also specific applications, problems, systems*)
Computer-aided instruction (CAI), 187
Computer vision, 19–25
Consonants, 123–124, 190
Content words, 210–216
Contextual analysis, 79, 80, 63–64, 227
  evaluation of, 58–59
  font design and, 73–74
  operating system and, 152
  phonetic decoding and, 143
  standardization of, 103, 233
Contours, of characters, 173–175
Control systems, 25–26, 145
Converted systems, 151
CRT display, 38–39, 69
Cursor positioning, 9, 40, 157

Daisy wheel printer, 42, 83–87
DALI system, 228
Database management system, 164–167
Dawaween, 159, 161, 162, 165–167
Decision support systems, 29
Devangari language, 7–16, 72–73
Development, computers and, 3–6
Diacritical marks, 80–84, 152
  display units and, 63
  drawing model and, 178–179
  frequency of occurrence, 98, 190–200
  Indian languages, 7
  keyboard design and, 64–65, 101
  phonetic transcription and, 132
Dictionary (*see* Lexicon)
Digitization methods, 40, 172–175
  resolution and, 70–77
  voice technology and, 43
Diphonemes, frequency of, 190
Disk storage, 36–37
Display systems, 38–39, 44, 62–63
  bilingual applications, 57–59, 229
  digitization and, 70–77
  full-screen editing, 157
  Indian languages and, 8–9
Document editors, 30–31
Dot matrix printing, 9
DRAGON system, 22
Drawing devices, 40–41
Drawing model, 178–179
Drum printers, 42
Dubbing, 151, 161
Dvorak-Dealey keyboard, 40
Dynamic programming, 26, 140–142, 144

EBCDIC code, 228
Editing systems:
  document editors, 30–31
  file editors, 233
  full-screen editing, 157
  proof applications, 186–187
  (*See also* Word processing systems)
Education, computer-aided, 187
Efficiency criteria, 107
8-bit codes, 103–105, 152
Electronic mail, 31, 35, 169
Electronic speech systems, 119–137, 138–147, 188
Electrostatic printers, 42–43
Engraving processes, 72–73, 93–94
Ergonomics, 40
Escape sequence, 103, 228
Expert systems, 19–28, 141

Feature vector, 113, 139–140
File editor, 233
File names, 110–111
5-bit codes, 102, 104
Floppy disks, 36
Folding criteria, 111–112
Fonts, printing:
  calligraphic definitions, 93–96
  classification of, 172
  design of, 69–77

Fonts, printing (*Cont.*):
graphic fonts, 172–175
logo fonts, 179–181
matrix-bound, 176–177
raster font, 176
(*See also* Printing systems)
Foreign word analysis, 216
Formants, 120, 125–126
Formatting:
of Arabic text, 177–182
for Indian languages, 9
word frequency and, 218
FORTRAN, 161
Frequency analysis:
of Arabic characters, 65, 88–90, 97–101, 190–200
of Arabic words, 218–221
for Indian languages, 8
Fricatives, 125
Full-screen editing, 157
Function words, 202–203

Glides, 123
Grammar:
machine translation system and, 234–237
syntactical analysis and, 152, 186, 201–207
word order, 143
Graphics:
graphic fonts, 172–175
software and, 151
(*See also* Display systems; Printing systems)

Ha al-ta nit character, 109
Hamza, 74, 110–112, 131, 152, 157
Harakat (*see* Diacritical marks)
Hardware, 61–62
Arabic systems and, 66, 67, 151, 153
bilingual systems and, 56–57
built-in limitations of, 225
machine translating system, 236
for work stations, 36–43
(*See also specific applications, systems*)
HARPY systems, 22, 26, 144
HEARSAY II system, 23, 24, 144
Hierarchical structures:
character recognition and, 113–114
communications and, 169–170
office systems and, 30, 33–34
speech recognition and, 144
Hot-metal typeface, 69

IBM computers, 56–57
Imports, 4
India, computers and, 3–6
Indian languages, 7–16, 72–73
Information processing, 19–28
(*See also specific applications, problems*)
Infrastructure, indigenous, 3–6
Ink jet printers, 42
Input/output systems (*see specific applications, systems*)
Inscriptional models, 72
Interactive processing, 226
Interconnect model, 33
Interfaces, 226–229
arabization software and, 161
input/output subsystem and, 36
telecommunications systems and, 169–170
(*See also* Communications systems)
Intonation, 132–136

Joystick, 41
Justification, of Arabic lines, 181

KATIB system, 74, 111
Keyboards, 39–41
bilingual, 104–105
character frequencies and, 89, 97–101
design principles for, 64–65
evaluation of, 58
for Indian languages, 8
standardization of, 104–105, 233
technology of, 39–41
Khawarizmi, 154, 157, 158
Koran, character frequency in, 194
Koufi script, 82, 86, 95, 176
Kursi, 111–112
Kuwait Institute for Scientific Research (KISR), 56

Lam-alef character, 74, 77, 114, 157
Language:
computer-aided instruction of, 187
language converter, 153–160
programming languages, 45, 46, 188–189, 225
understanding of, 21–22
(*See also* Grammar; *specific problems*)
Larynx, 119
Laser printers, 42–43
Layered systems:
Arabic character set for, 110–112
communications and, 169–170
office systems and, 33–34
sufficiency and, 107–112
Lexicon:
of affixes, 203
data base for, 185
machine translation and, 235–236
morphological analysis and, 212

Lexicon (*Cont.*):
proofing applications, 186–187
speech synthesis and, 129
Ligature, 7, 74, 80, 114, 171, 179
Light emitting diode (LED), 39, 232
Light pen, 41
Linear predictive coding (LPC), 124, 126
Line-feed system, 169
Line justification, 178, 181
Line printers, 90–94
Linguistics, impact on informatics, 185–189
(*See also specific applications, problems*)
Liquid crystal display (LCD), 39
Logo fonts, 179–181

Machine language, 45, 153
Machine translation, 187, 234–237
Macro applications, 229
Mail systems, 31, 35, 169
Management systems, 4, 29–51
Markov process, 145
Matrix-bound fonts, 176–177
Matrix size, 8, 63
(*See also* Display systems)
Medical applications, 20
Melodic curve, 132–136
Memory, 25, 36–37, 38
Menus, 49
Mesa system, 46
METAFONT system, 71–76
METAL system, 234–237
Microcomputers, evaluation of, 59–61
(*See also specific applications*)
Microprocessors, 36, 57
Mim character, 76
MIRABELLE system, 23
Monitor systems (*see* Display systems)
Morphological analysis, 201–207
comprehensive generator for, 208–217
context sensitivity and, 73–74
lexical data base and, 185
logo fonts and, 179–181
Mouse, 41
Multilingual systems, 8–9
(*See also* Bilingual systems)
Munasseq Alkalimatt, 164
MYCIN system, 19–20

Nagari language, 73
Najla, 160
Names, 90
Nasals, 123, 125
Naskh style, 81, 87, 95, 173, 175
Negation particle, 114
Networks, 31, 169–170
(*See also* Communications systems)
Nirnaya Sagar, 72
Nominal structure, 132
Noun forms, 210–211, 219
Numerals, 78, 88, 139, 141
Nun character, 77

Object-oriented languages, 25
Office systems, 29–51, 88
Operating systems, 45–46, 226–228
arabization of, 66–67, 152–153, 154
evaluation of, 61
systems utilities and, 161
Optical disks, 37–38
Optical document entry, 188
Overlap, of Arabic characters, 179
Overlaying processes, 177

Parsing, 186, 235
PASCAL, 100, 161
Persian, 107, 109
Pharyngealization, 131, 143
Phonemes, 123, 128, 143–144, 190
Phonetics:
Arabic system of, 123–124, 143–144
orthographic transcription and, 130–132
phonetic decoding, 143–144
production rules and, 24
speech synthesis and, 128
Phosphor, 38
Photography, 24, 69, 172
Picture description language, 115
Pilot system, 46
Pitch analysis, 24, 43, 132
Pixel, 38
(*See also* Display systems)
Plasma panels, 39
Point detection, 115–116
Pointing devices, 40–41
Predicates, rule systems and, 206
Prefixes, 201, 203, 209–217, 218–221
Primitives, character recognition and, 113–118
Printing systems:
bilingual applications, 57, 102–103, 228–229, 231–232
calligraphic model, 72–77, 93–96
character frequency and, 89
character repertoire and, 91–92
daisy wheel, 42, 83–87
evaluation of, 65–66
Indian languages and, 7, 9, 72–73
performance optimization, 90–94
speed of, 92–93
technology for, 41–43

Printing systems (*Cont.*):
teleprinter, 102–103
(*See also* Fonts; *specific systems*)
Procedural attachment, 25
Production rule paradigm, 23–24
Programming languages, 45, 46, 188–189, 225
Program translators, 153, 157–161
Project INTERACT, 3–6
Proofing applications, 186–187
Prosody, 127, 128, 144
Protocols, 3

Qaf character, 77
Query system 187
Quran, character frequencies in, 194

Ra character, 76
Rasters, 38, 44, 176
RCTC (*see* Research Computer Technology Corporation)
Recognition, of characters, 113–118, 188
Relaxation techniques, 26
Research Computer Technology Corporation (RCTC), 153, 156, 163
Resolution, 69–77
Roukai style, 80
Rule-based systems:
machine translation and, 234–237
morphological analysis and, 201–207
production rules and, 23–24
speech synthesis and, 128
syntax parsing and, 186

Sanskrit, 72
SAUDIA language, 161
Screen masking, 232
Search methods, 26, 145
Segmentation:
Arabic characters and, 152
character recognition and, 113–118
overlapping and, 179
speech recognition and, 138
suprasegmental phenomena, 143–144
Semantic analysis, 143, 186
Semi-vowels, 123
Sentences:
digital storage of, 127, 129
generation of, 186
speech synthesis and, 127
structures of, 132–136
7-bit codes, 102–105, 152–153
Social structures, 30–31
Software, 61, 151–168
arabization of, 66–67, 229
bilingual applications, 231
built-in limitations of, 225
classification of, 151
communications systems and, 169–170
database management and, 164–167
programming languages, 45
program translators, 153, 157–161
word processing and, 161–164
Sorting, 111, 139, 218
Spectrogram interpretation, 24
Speech recognition, 19–26, 43–44, 119, 138–147, 188
Speech synthesis, 43, 119–137, 188
Spelling, 186–187
SPOOL-SE system, 228–229
Spreadsheets, 31, 165
Standardization, 233
for Arabic characters, 79–80, 102–106, 107–112
ASMO and, 55, 102, 107–112, 153, 155
survey for, 57–66
(*See also* Character repertoire; *specific problems, systems*)
Statistical analysis:
characters frequencies, 88–90, 97–101, 190–200
speech recognition and, 141
word frequencies, 218–221
Stops, 123
Storage systems, 36–37, 39
Structural words, 210, 212–213
Sub-alphabetic characters, 109, 111
Sufficiency criteria, 107–112
Suffixes, 201, 203, 209–217, 218–221
Supra-glottal tract, 119
Suprasegmental phenomena, 143–144
Switching technology, 39–41
Syllables, 144
Syntax, 152, 201–207
character recognition and, 113–118
parsing for, 186
speech recognition and, 143
Syrian Scientific Studies and Research Center (SSRC), 56, 102

Ta character, 76–77, 109
Tamil language, 73
Tansiiq Alkalimaat, 164
Tasdid character, 111
Telecommunications, 36, 102–103, 169–170
(*See also* Communications systems)
Television, 38
Telugu language, 7–16
Text formatting, 177–182
Thermal printers, 42
Thouloth style, 81, 85

Time normalization, 139–141
Timesharing, 35
Top-down control systems, 25
Touch pads, 40–41
Transcription, to phonetics, 130–132
Translation systems,153, 157–160, 187, 234–237
Turbo-Pascal, 161, 162
Typefaces, design of, 69–77
  (*See also* Fonts; Printing systems)
Typewriter, 40
Typing speed, 97

Understanding systems, 19–28, 142–145, 185–189
UNIX system, 44, 46

Vacuum fluorescent display, 39
Vector methods, 38–39, 174–175
Verb forms, 210, 219
Video display (*see* Display systems)
VISICALC, 165
Viterbi algorithm, 144
Vocal apparatus, 119–122
Voiced sounds, 121, 123
Voice technology, 43–44, 124
Vowels, 111, 112, 125, 190–191
  frequencies of, 190
  Indian languages and, 8
  long vowels, 131
  voicing of, 121, 123

Winchester disk, 36
Window systems, 46–49, 164
Word processing systems, 61, 161–164
  file editing and, 233
  multilingual, 8–9
  office systems and, 29
  text formatting, 177–182
Words:
  content words, 210–216
  frequency table of, 218–221
  lexical database, 185
  parts of, 113–118
  recognition system for, 139
  speech synthesis of, 127–128
  structural, 210
  text formatting and, 178
  word spotting, 144
WORDSTAR, 164
Workstations, 30, 35–43

XCON system, 20